Rozvrhování s omezujícími podmínkami (dokončení)

21. března 2022

Alternativní zdroje

Jak modelovat alternativní zdroje pro danou aktivitu?

Pro každý zdroj uděláme duplikát aktivity

- duplikát se účastní příslušných zdrojových podmínek,
  ale neomezuje další aktivity na daném zdroji
  - „neúspěch" u duplikátu znamená odstranění zdroje z domény proměnné resource(A) příslušné aktivity
  - odstranění zdroje z domény proměnné resource(A) znamená „smazání" odpovídajícího duplikátu
- původní aktivita se účastní precedenčních podmínek
  (např. v rámci multi-operační úlohy)
- omezení časů u duplikátu se propaguje do originálu aktivity a naopak

Alternativní zdroje: odvozovací pravidla

Nechť A_u reprezentuje duplikát aktivity A na zdroji u∈resource(A), pak probíhají následující propagace:

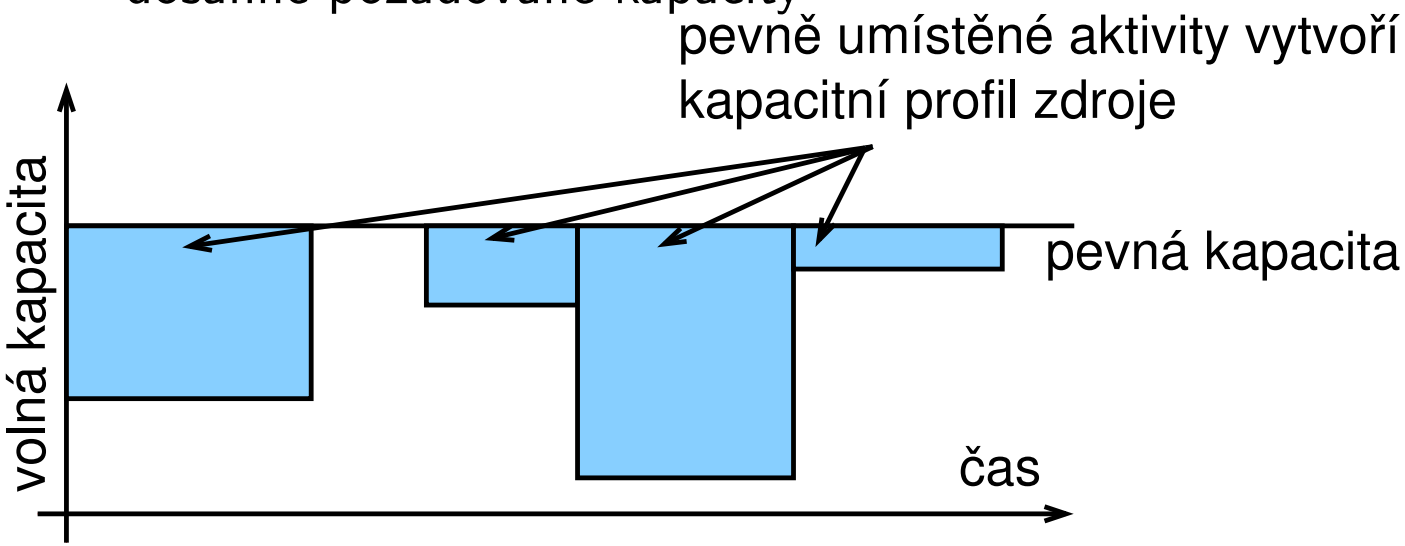
- u∈resource(A) $\Rightarrow$ start(A) $\leq$ start(A_u)
- u∈resource(A) $\Rightarrow$ end(A_u) $\leq$ end(A)

- start(A) $\geq$ min{start(A_u): u $\in$ resource(A)}
- end(A) $\leq$ max{end(A_u): u $\in$ resource(A)}

- neúspěch pro A_u $\Rightarrow$ resource(A)\{u}

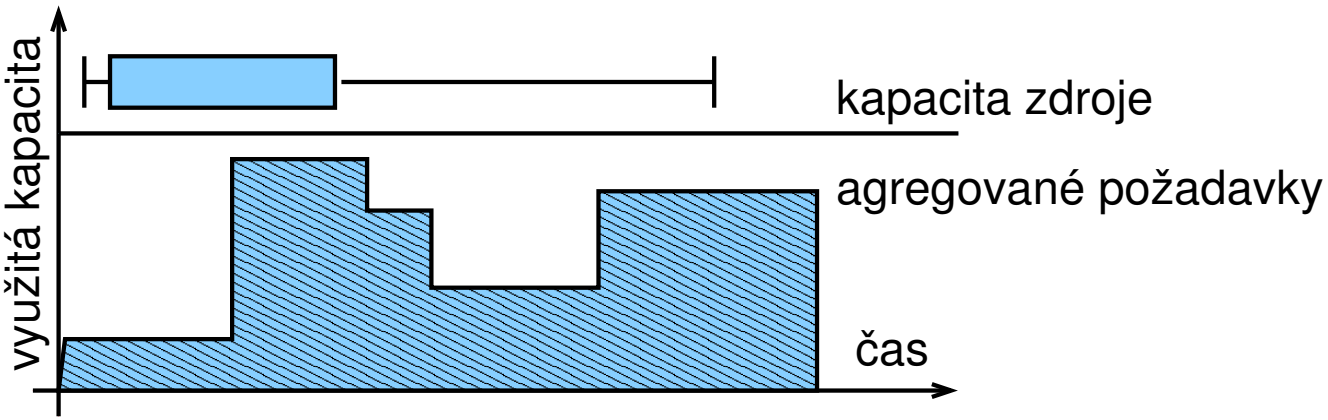
Kumulativní zdroje

- Každá aktivita využívá jistou kapacitu zdroje cap(A)
- Aktivity mohou být zpracovány paralelně, pokud není překročena kapacita zdroje
- Kapacita zdroje může být v čase proměnná
  - takové zdroje lze modelovat pomocí v čase neměnné kapacity, od které se odečte kapacita pevně umístěných aktivit, čímž se v každém čase dosáhne požadované kapacity

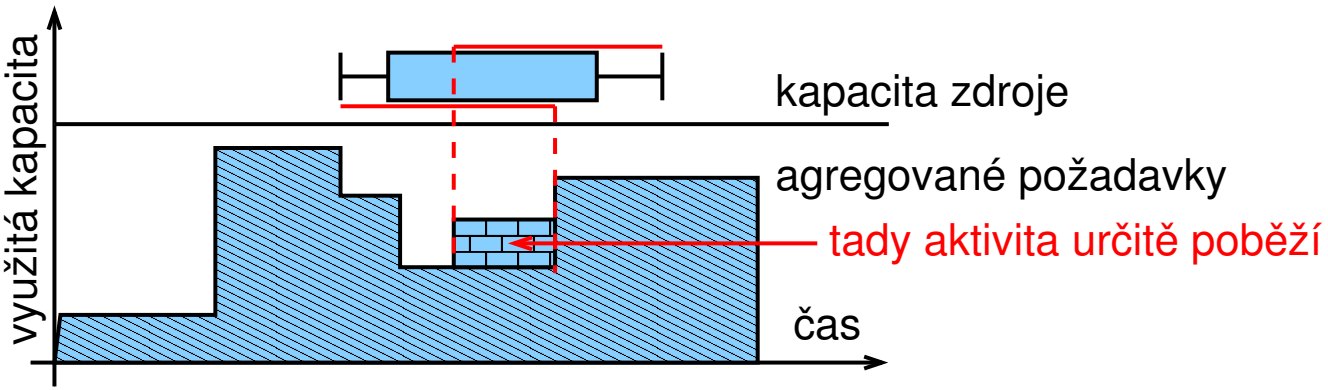


Agregované požadavky

- Baptiste et al. 2001
- Kdy je dostatečná kapacita pro zpracování aktivity?



- Jak se konstruuje graf agregovaných požadavků?



Podmínka tabulky *(timetable constraint)*

- Uvažujeme diskrétní čas
- Jak zajistit, že v žádném čase není překročena maximální kapacita?

$$\forall t \sum_{start(A_i) \leq t \leq end(A_i)} cap(A_i) \leq MaxCapacity$$

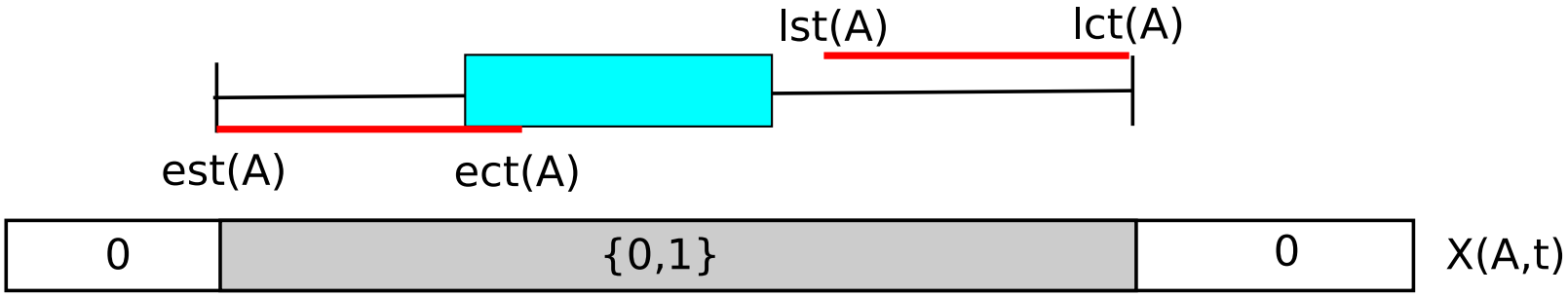
- Tabulka *(timetable)* pro aktivitu A je množina boolovských proměnných $X(A, t)$ udávajících, zda A běží v čase t

$$\forall t \sum_{A_i} X(A_i, t) cap(A_i) \leq MaxCapacity \qquad (*)$$

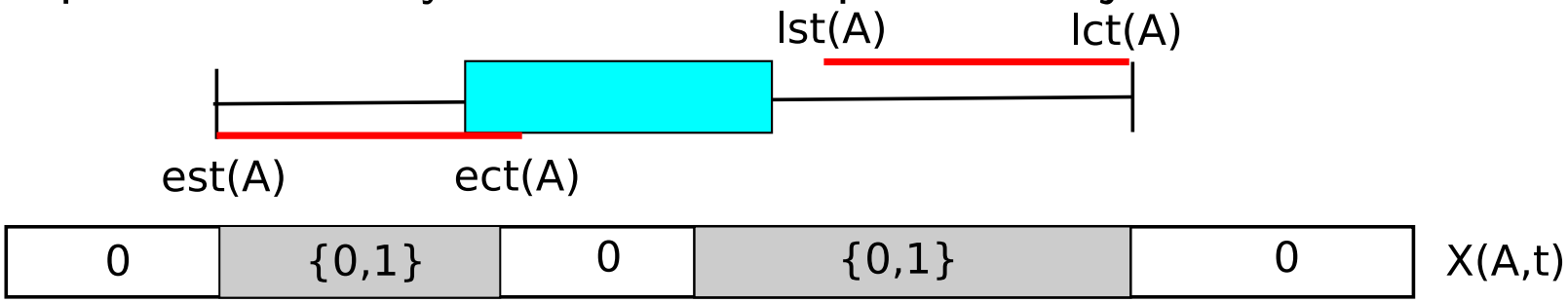
$$\forall t, i \; start(A_i) \leq t \leq end(A_i) \Leftrightarrow X(A_i, t)$$

Podmínka tabulky: př. s odvozovacími pravidly

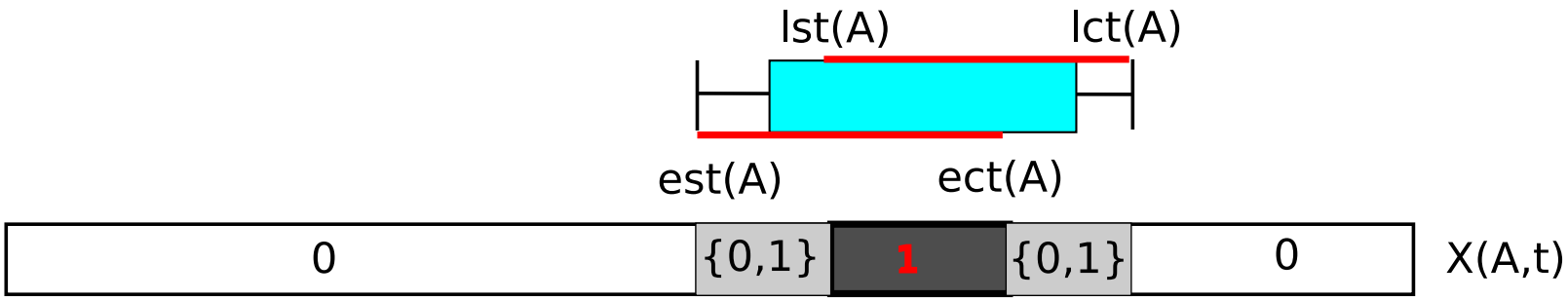
Počáteční stav



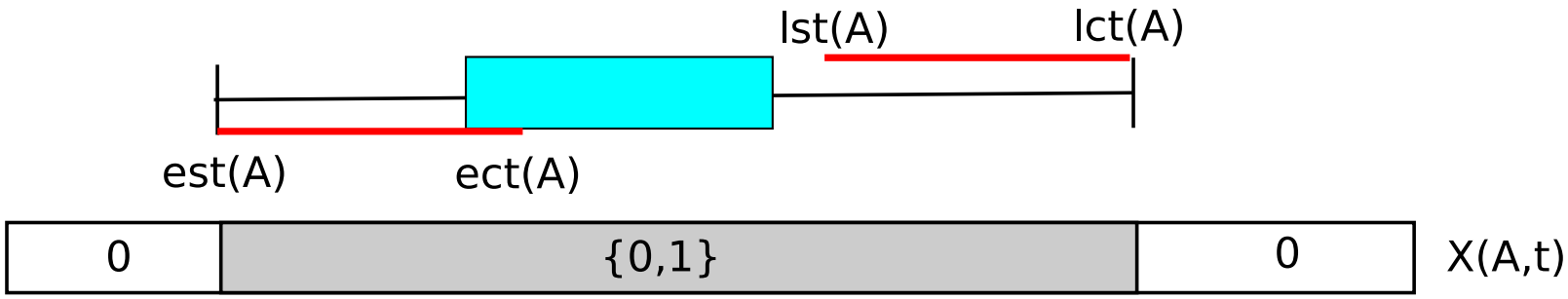
Některé pozice zakázány vzhledem ke kapacitě zdroje



Nový stav



Podmínka tabulky: odvozovací pravidla

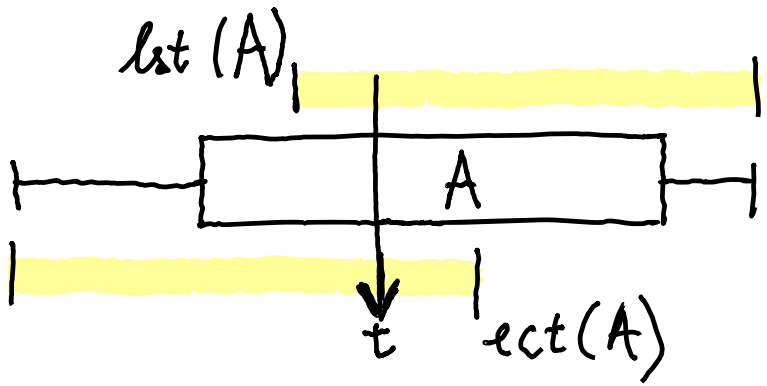


Jak realizovat filtraci přes omezení

$$\forall t, i \; start(A_i) \leq t < end(A_i) \Leftrightarrow X(A_i, t) \; ?$$

Problém: t je zároveň index a také proměnná

- start(A) $\geq$ min{t | 1 $\in$ X(A,t)}
- end(A) $\leq$ 1+max{t | 1 $\in$ X(A,t)}
- X(A,t)=0 $\wedge$ t < ect(A) $\Rightarrow$ start(A)>t
- X(A,t)=0 $\wedge$ lst(A)$\leq$t $\Rightarrow$ end(A)$\leq$t
- lst(A)$\leq$t $\wedge$ t < ect(A) $\Rightarrow$ X(A,t)=1



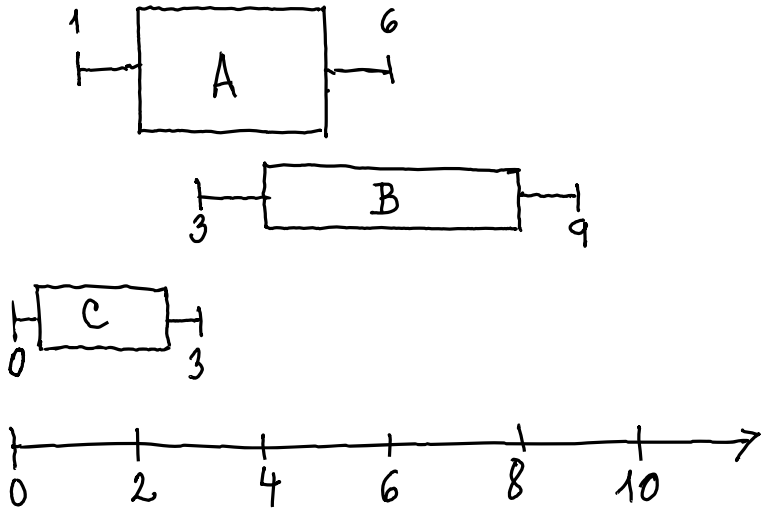
Cvičení: podmínka tabulky

Máme zadány zdroj s kapacitou 2 a aktivity

j	cap(j)	est(j)	lct(j)	p(j)
A	2	1	6	3
B	1	3	9	4
C	1	0	3	2

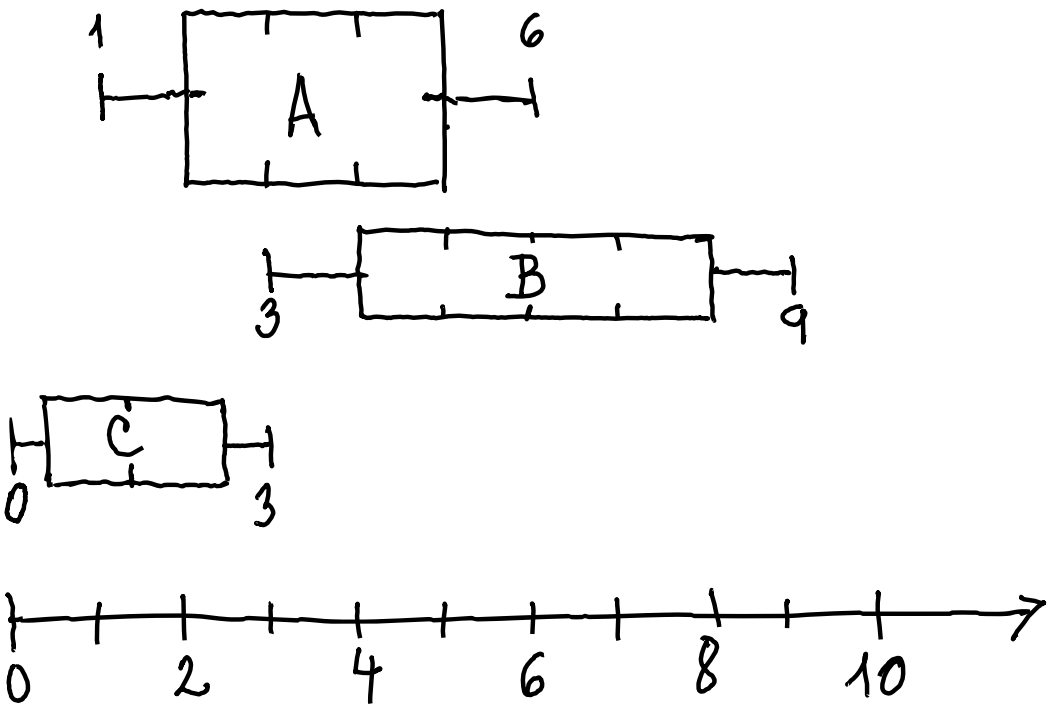
1. Jak jsou inicializovány proměnné X(j,t)?
2. Jak se jejich hodnoty mění při použití odvozovacích pravidel podmínky tabulky?
3. Jak by mohly vypadat výsledné rozvrhy po aplikaci pravidel?

Cvičení: podmínka tabulky



1. Jak jsou inicializovány proměnné X(j,t)?
   - $X(A,1)$ až $X(A,5)$ jsou $\{0,1\}$, $X(B,3)$ až $X(B,8)$ jsou $\{0,1\}$, $X(C,0)$ až $X(C,2)$ jsou $\{0,1\}$, ostatní proměnné nulové
2. Jak se jejich hodnoty mění při použití odvozovacích pravidel podmínky tabulky?
   1. dle $(*)$ B může začít nejdříve v čase 4 kvůli A, tj. $X(B,3)=0$ a A musí být před B, tj. A nejpozději skončí v čase 5 a $X(A,5)=0$
   2. dále z $(*)$ $X(C,2)=0$, C začne v čase 0
      $X(A,1)=0$ a A začne v čase 2 a také
      B musí začít až v čase 5 a $X(B,4)=0$
      a máme jediné řešení

Cvičení: podmínka tabulky

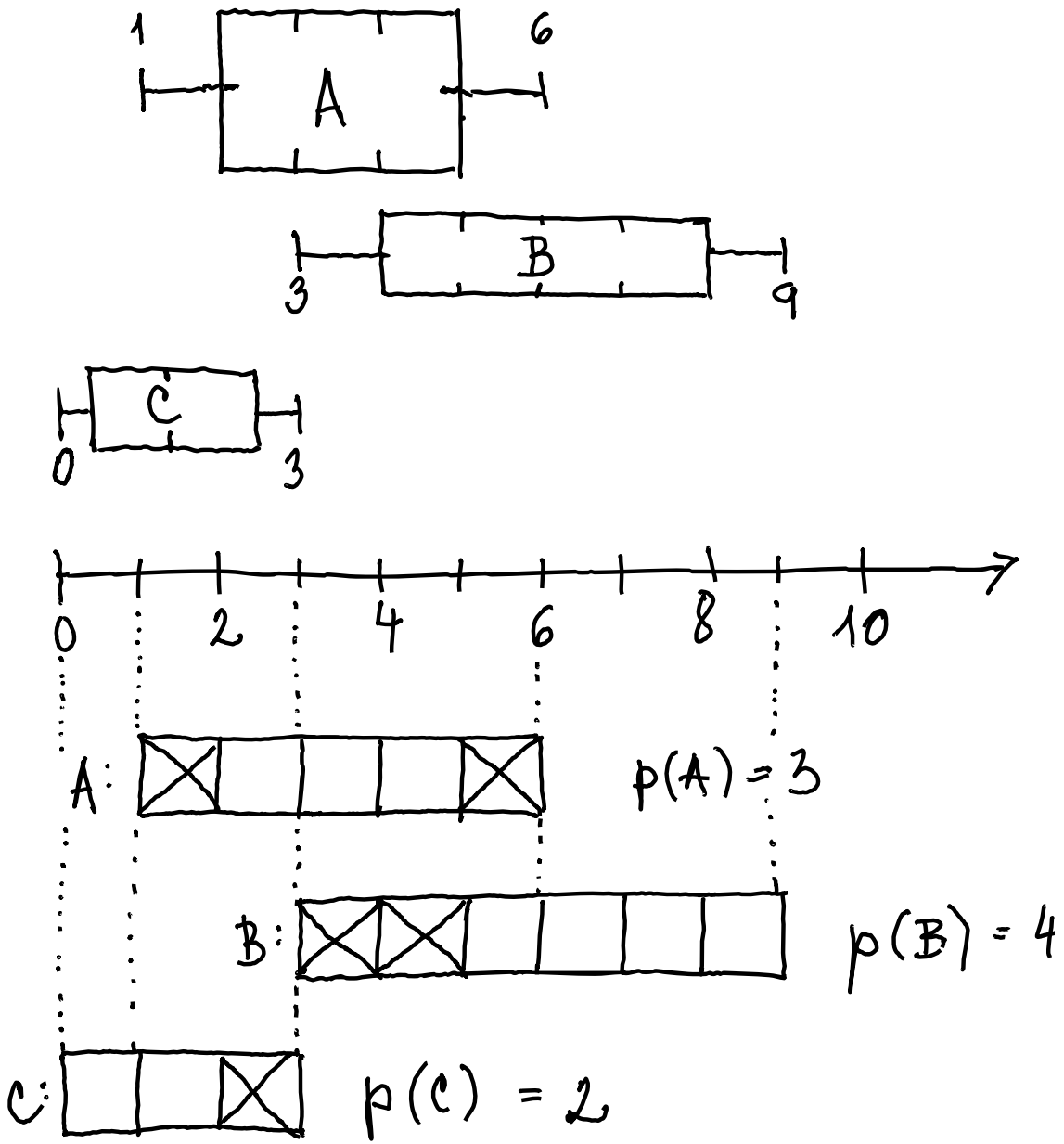


A: $p(A) = 3$

B: $p(B) = 4$

C: $p(C) = 2$

Cvičení: podmínka tabulky



Unární a kumulativní zdroje: shrnutí a možná rozšíření

Disjunktivní omezení

- známe: unarní zdroje, nepřerušitelné aktivity
- rozšíření: přerušitelné aktivity, kumulativní zdroje

Hledání hran

- známe: unarní zdroje, nepřerušitelné aktivity
- rozšíření: přerušitelné aktivity, kumulativní zdroje

Ne-první/ne-poslední

- známe: unarní zdroje, nepřerušitelné aktivity
- rozšíření: kumulativní zdroje

Podmínka tabulky

- známe: kumulativní zdroje, nepřerušitelné aktivity
- rozšíření: přerušitelné aktivity

Globální podmínky

Pro reprezentaci zdrojů využívány v programovacích jazycích
tzv. globální podmínky

- definované pro libovolný konečný počet proměnných
- komplexní podmínky s vlastním propagačním algoritmem

Základní globální podmínky (pro rozvrhování)

- příklady z IBM ILOG OPL (Optimization Programming Language)
- všechny proměnné různé
  - `allDifferent`
- disjunktivní zdroj
  - `dvar interval`, `dvar sequence`
  - `noOverlap`
- kumulativní zdroj
  - `cumuFunction`, `pulse`

Všechny proměnné různé

Proměnné v poli `Array` jsou různé

- reprezentace unárního zdroje s jednotkovou dobou trvání všech aktivit
- `dvar int Array[Interval];`
- globální podmínka: `allDifferent(Array)`

Příklad: učitelé musí učit v různé hodiny

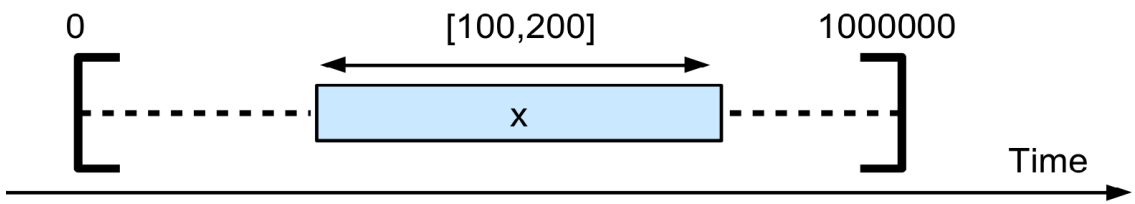
- `Jan = 6, Ota = 2, Anna = 5,`
  `Marie = 1, Petr` $\in \{3,4\}$`, Eva` $\in \{3,4\}$

učitel	min	max
Jan	3	6
Petr	3	4
Anna	2	5
Ota	2	4
Eva	3	4
Marie	1	6

Intervalové proměnné

Intervalová proměnná: pro modelování časového intervalu (úlohy, aktivity)

- hodnotou intervalové proměnné je celočíselný interval $[start, end)$
- příklad: `dvar interval x in 0..1000000 size 100..200;`



Disjunktní rozvrhování

Sekvenční proměnná `p`

- definována na množině intervalových proměnných `x`

```
dvar interval x[i in 1..n] ...;
dvar sequence p in x;
```

- hodnota intervalové proměnné `p` je permutace přítomných intervalů
  - pozor, permutace `t` ještě neimplikuje žádné uspořádání v čase

Omezení `noOverlap(p)`

- vyjadřuje, že sekvenční proměnná `p` reprezentuje řetězec nepřekrývajících se intervalových proměnných
- pro vyjádření rozvrhování na unárním/disjunktivním zdroji, kde se intervaly/úlohy nepřekrývají

Precedence, účelová funkce

Mezi intervalovými proměnnými můžeme definovat precedenční podmínky:

```
dvar interval i;
dvar interval j;


endBeforeStart(i, j);
startBeforeStart(i, j);
startAtStart(i, j);
...
```

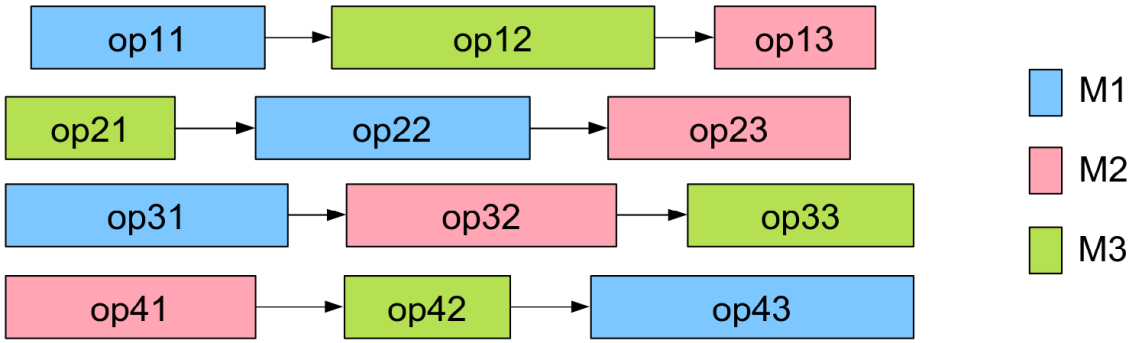
Pro vytváření účelových funkcí nebo definici omezení

```
startOf(x)
endOf(x)
sizeOf(x, V)
```

Příklad: minimalizace makespanu

```
minimize max(i in 1..n) endOf(x[i])
```

Příklad: rozvrhování problému job-shop




```
tuple Operation {
  int mch; // machine
  int pt;  // processing time
};
Operation Ops[j in Jobs][p in Pos] = ...;
```

```
1  dvar interval op[j in Jobs][p in Pos] size Ops[j][p].pt;
2  dvar sequence mchs[m in Mchs] in
3    all(j in Jobs, p in Pos: Ops[j][p].mch == m) op[j][p];
4
5  minimize max(j in Jobs) endOf(op[j][nbPos]);
6  subject to {
7    forall(m in Mchs)
8      noOverlap(mchs[m]);
9    forall(j in Jobs, p in 2..nbPos)
10     endBeforeStart(op[j][p-1],op[j][p]);
11 }
```

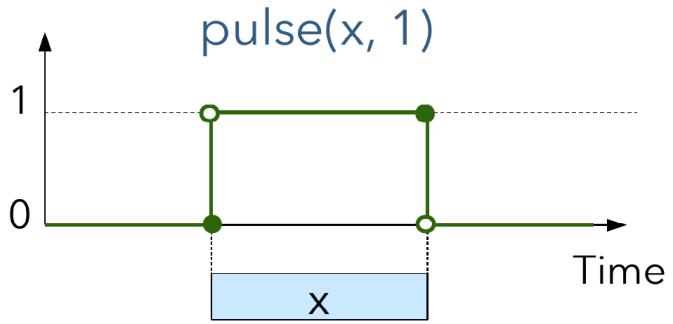
`op[j][p]` odkazuje operaci úlohy `j`, která je zpracovávána v rámci úlohy jako p-tá

Kumulativní zdroj pomocí kumulativní funkce

Hodnota výrazu kumulativní funkce reprezentuje vývoj kvantity v čase, která může být inkrementálně změněna (snížena nebo navýšena) intervalovými proměnnými.

Intervalové proměnné `x[i]` přispívají do kumul. funkce po dobu svého provádění

```
int capacity[1..5] = [1,3,2,4,1];
cumulFunction y = sum(i in 1..5) pulse(x[i],capacity[i]);
```



Omezení na výrazech kumulativní funkce: pro omezení kapacity zdroje

```
int h = ...
cumulFunction f= ...
f<=h
```

Příklad: job-shop a omezení celkového počtu strojů

```
cumulFunction allMachines = sum(j in Jobs, p in Pos) pulse(op[j][p],1);
allMachines <= m;
```

Prohledávání/přiřazování (opakování)

Konzistenční techniky jsou (obvykle) neúplné

$\Rightarrow$ potřeba prohledávací algoritmus, který vyřeší "zbytek"

Přiřazovaní *(labeling)*

- prohledávání do hloubky (DFS/BT)
  - přiřaď hodnotu do proměnné
  - propaguj = uděl̆ej
    problém lokálně konzistentní
  - vrať se v případě neúspěchu

Způsoby větvení

Jaká proměnná má být ohodnocena první?

- princip prvotního neúspěchu *(first-fail)*
  - preferuj proměnnou, jejíž přiřazení je nejobtížnější
  - např. proměnné s nejmenší doménou:
    doména se snadněji vyprázdní
  - nebo proměnné s nejvíce podmínkami: pro proměnné s více podmínkami
    je obecně obtížnější nalézt hodnotu proměnné
- definuje tvar prohledávacího stromu
  - výběr proměnné s malou velikostí domény: malé větvení na této úrovni
  - výběr proměnné s velkou velikostí domény: velké větvení na této úrovni

Jaká hodnota má být vyzkoušena první?

- princip prvotního úspěchu *(succeed-first)*
  - preferuj hodnoty, které nejspíše patří do řešení
  - např. hodnoty s nejvíce podporami v okolních proměnných
  - tato heuristika je obvykle problémově závislá
- definuje pořadí procházení větví

Schémata větvení

Větvení = řešení disjunkcí

Tradiční rozvrhovací přístupy

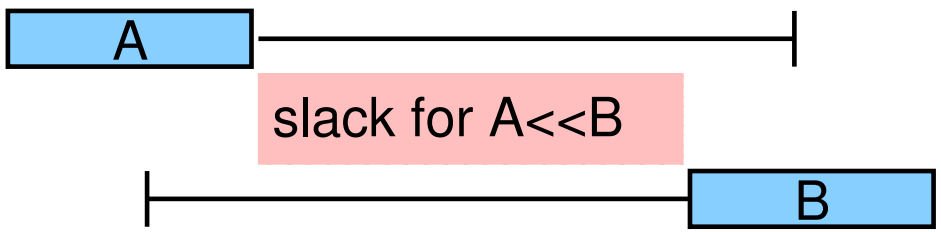
- kritická rozhodnutí se dělají první
  - vyřeš kritická místa *(bottlenecks)*, ...
  - definuje tvar prohledávacího stromu
  - podobně jako princip prvního neúspěchu *(first-fail)*
- preferuj alternativy s větší flexibilitou
  - definuje pořadí větví pro prozkoumání
  - podobně jako princip prvního úspěchu *(succeed-first)*

Jak popsat, co je kritické a co je flexibilní?

Rezerva *(slack)* Smith&Cheng 1993

- Rezerva *(slack)* je formální popis flexibility
- Rezerva pro dané pořadí dvou aktivit

  „volný čas pro posunování aktivit"



$slack(A << B) = \max(end(B)) - min(start(A)) - p(A) - p(B))$

- Rezerva pro dvě aktivity (bez určení pořadí)

  $slack(\{A, B\}) = \max(slack(A << B), slack(B << A))$

- Rezerva pro skupinu aktivit

  $slack(\Omega) = \max(end(\Omega)) - min(start(\Omega)) - p(\Omega)$

Větvení uspořadáním dvojic aktivit

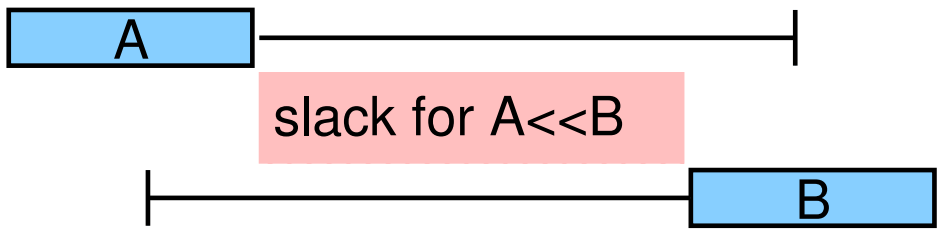
$A << B \quad \vee \quad \neg A << B$

Jaké aktivity mají být uspořádány první?

- nejkritičtější pár (first-fail)
- pár s minimální rezervou $slack(\{A, B\})$

Jaké pořadí aktivit má být zvoleno?

- nejflexibilnější pořadí (succeed-first)
- pořadí maximalizující $slack(A??B)$
- Příklad: vybereme pořadí $A << B$



Bodů volby při n aktivitách: $O(n^2)$

Větvení první/poslední — Baptiste at al. 1995

$$(A << \Omega \vee \neg A << \Omega) \quad \vee \quad (\Omega << A \vee \neg \Omega << A)$$

Máme hledat první nebo poslední aktivitu?

- díváme se na množinu možných kandidátů na první aktivitu a na množinu možných kandidátu na poslední aktivitu
- výbereme menší z těchto dvou množin (first-fail)
  - menší počet kandidátů znamená, že je těžší najít vhodného kandidáta

Jaká aktivita má být vybrána?

- pokud se hledá první aktivita, potom preferuj aktivitu, která má nejmenší $min(start(A))$
- pokud se hledá poslední aktivita, potom preferuj aktivitu, která má největší $max(end(A))$

Bodů volby: $O(n)$

Zdrojová rezerva

Zdrojová rezerva je definovaná jako

rezerva množiny aktivit zpracovávaných daným zdrojem

Jak používat zdrojovou rezervu?

- pokud volíme zdroj, na kterém budou aktivity uspořádány jako první
  - vyber zdroj s minimální rezervou (kritické místo)
- pokud volíme zdroj, na který alokovat danou aktivitu
  - vyber zdroj s maximální rezervou (flexibilita)

Omezující podmínky: shrnutí

Problém splňování podmínek

- popis problému pomocí doménových proměnných a omezení
- konzistence a propagace
- prohledávání

Rozvrhování jako problém splňování podmínek

- doménové proměnné pro čas a zdroje
- propagační algoritmy pro
  - unární zdroje
  - kumulativní zdroje
  - produkovatelné a spotřebovatelné zdroje
  - alternativní zdroje
- globálních podmínky pro zdroje
- prohledávání a rozvrhovací strategie
  - pojem rezervy
  - přístupy k větvení