CHAPTER IV

FLOW SHOP SYSTÉMY

Doteraz sme predpokladali, že každá úloha pozostávala z jedinej operácie. Prejdime teraz k rozvrhovacím problémom, v ktorých sa každá úloha J_i bude skladať z m operácií, t.j. $J_i = \{o_{i1}, o_{i2}, \ldots, o_{im}\}$, pričom pre každú úlohu J_i platí

$$o_{i1} \prec\prec o_{i2} \prec\prec \cdots \prec\prec o_{im}.$$

Tieto operácie prechádzajú strojmi v rovnakom poradí, t.j. operácie o_{i1} sa spracujú na stroji M_1, operácie o_{i2} na stroji M_2 atď. Jednotlivé úlohy J_i sa líšia len časmi spracovania p_{ij} operácie o_{ij} na stroji M_i. V Conwayovej klasifikácii pôjde o systémy $m|n|F|f$, v klasifikácii podľa LLRK $Fm||f$.

Existuje aj zložitejší pohľad na flow shop problémy, kedy úlohy J_i nemusia mať všetkých m operácií. Tieto problémy sa však vo väčšine prípadov dajú previesť na predchádzajúci tvar zavedením fiktívnych operácií s nulovým (alebo infinitezimálnym) časovým trvaním.

Budeme teda predpokladať, že:

- Je daných n úloh $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$, z ktorých má každá po m operácií prechádzajúcich strojmi v rovnakom poradí $M_1, M_2, \ldots, M_m$.
- Sú dané doby spracovania p_{ij} operácií o_{ij}. Nastavovacie časy operácií sú nezávislé na poradí a sú zahrnuté v časoch spracovania p_{ij}.
- Stroje $M_1, M_2, \ldots, M_m$ sú neustále k dispozícii.
- Individuálne operácie nie sú prerušiteľné.

Za týchto okolností je rozvrh určený poradím spracovania úlohy na každom stroji a takýchto poradí je

$$(n!)^m$$

VETA. *V systémoch $Fm||f$ kde f je regulárne kritérium stačí pri hľadaní optima uvažovať iba tie rozvrhy, pre ktoré je poradie prechodu úloh na prvých dvoch strojoch rovnaké.*

Dôkaz.

Keby nebolo poradie prechodu oboch úloh na prvých dvoch strojoch rovnaké, existovala by taká dvojica J_i, J_j bezprostredne za sebou idúcich úloh na prvom stroji, že na druhom stroji sa najprv vykoná J_j a potom J_i. Spracovanie úlohy J_j na druhom stroji môže začať až v čase, keď sa ukončí jej spracovanie na prvom stroji.

Prehodením spracovania poradia týchto úloh na prvom stroji sa vytvorí priestor pre skoršie spracovanie úlohy J_j na druhom stroji, pričom môže dôjsť k urýchleniu spracovania ostatných úloh. □

VETA. *V systémoch* $Fm||\mathbf{C_{max}}$ *stačí pri hľadaní optima uvažovať iba tie rozvrhy, pre ktoré je poradie prechodu úloh na posledných dvoch strojoch rovnaké.*

Dôkaz.

Keby nebolo poradie prechodu oboch úloh na posledných dvoch strojoch rovnaké, existovala by taká dvojica $J_i,\ J_j$ bezprostredne za sebou idúcich úloh na poslednom stroji, že na predposlednom stroji sa najprv vykoná J_j a potom J_i. Spracovanie úlohy J_j na druhom stroji môže skončiť skôr, než sa začneí jej spracovanie na poslednom stroji. Prehodením spracovania poradia týchto úloh na poslednom stroji sa neoneskorí čas ukončenia poslednej úlohy na poslednom stroji, ba môže sa vytvoriť priestor pre skoršie spracovanie úlohy J_j na druhom stroji, čo môže viesť i k urýchleniu spracovania poslednej úlohy na poslednom stroji. □

DÔSLEDOK. *V systémoch* $F3||\mathbf{C_{max}}$ *stačí pri hľadaní optima uvažovať iba tie rozvrhy, pre ktoré je poradie prechodu úloh na všetkých troch strojoch rovnaké.*

Johnsonov problém.

Najjednoduchším prípadom systému $Fm||\mathbf{C_{max}}$ je systém $F2||\mathbf{C_{max}}$. Úloha minimalizovať $\mathbf{C_{max}}$ v dvojstrojovom flow shop systéme je známa ako Johnsonov problém. Pretože ide o regulárne kritérium, poradie úloh musí byť na oboch strojoch rovnaké. Riešením bude teda nejaký permutačný rozvrh.

JOHNSONOVO PRAVIDLO. *V optimálnom rozvrhu pre systém* $F2||\mathbf{C_{max}}$ *úloha* J_i *predchádza úlohu* J_j*, ak*

$$\min\{p_{i1}, p_{j2}\} \leq \min\{p_{j1}, p_{i2}\}$$

Dôkaz tohoto tvrdenia je netriviálny.

JOHNSONOV ALGORITMUS.

KROK 1: Označme $\mathcal{I}$ množinu nezaradených úloh. Inicializačne položme $\mathcal{I} = \mathcal{J}$.

KROK 2: Nájdi $\min_{i\in\mathcal{I}}\{p_{i1}, p_{i2}\}$. Nech minimum nastáva pre úlohu J_j

KROK 3: Ak minimálny čas spracovania vyžaduje stroj M_1 zaraď úlohu J_j na prvé voľné miesto, inak zaraď úlohu J_j na posledné voľné miesto.

KROK 4: Položme $\mathcal{I} = \mathcal{I}\ - \{J_j\}$. Ak $\mathcal{I} = \varnothing$ STOP, inak goto KROK 2.

Pre systémy $F3||\mathbf{C_{max}}$ optimálny rozvrh je stále ešte permutačným rozvrhom, ale nemáme všeobecný polynomiálny algoritmus na jeho hľadanie. V niektorých špeciálnych prípadoch však existuje optimálny postup.

VETA. *Nech platí niektorá z podmienok:*

$$p_{i2} \leq p_{j1} \quad \forall i \neq j$$

$$p_{i2} \leq p_{j3} \quad \forall i \neq j$$

$$p_{i2} \leq \min(p_{i1}, p_{i3}) \quad \forall i \ .$$

Potom v optimálnom rozvrhu pre systém $F3||\mathbf{C_{max}}$ *úloha* J_i *predchádza úlohu* J_j*, ak*

$$\min\{p_{i1} + p_{i2}, p_{j2} + p_{j3}\} \leq \min\{p_{i2} + p_{i3}, p_{j1} + p_{j2}\}.$$

Vieme tiež, že ak pre dvojstrojové problémy s časmi spracovania $\{p_{i1}, p_{i2}\}$ a $\{p_{i2}, p_{i3}\}$ dostaneme rovnaké optimálne rozvrhy, potom je toto poradie úloh optimálne i pre trojstrojový problém s časmi spracovania $\{p_{i1}, p_{i2}, p_{i3}\}$.

Všeobecný flow shop problém.

Všeobecný flow shop problém $Fm||\mathbf{C_{max}}$ je NP–ťažkou úlohou. Ako už bolo spomenuté, ide o vybratie optimálneho riešenia z $(n!)^{(m-2)}$ možností v prípade kritéria $\mathbf{C_{max}}$, resp. $(n!)^{(m-1)}$ možností v prípade iného regulárneho kritéria.

Uvedieme niekoľko heuristických postupov na jeho riešenie. Všetky heuristiky vytvárajú len permutačné rozvrhy (t.j. rozvrhy, v ktorých je poradie spracovania úloh na všetkých strojoch rovnaké.

Palmerova heuristika pre flow shop $Fm||\mathbf{C_{max}}$.

KROK 1: Pre $j = 1, 2, \ldots, n$ vypočítajme

$$s_j = (m-1).p_{jm} + (m-3).p_{jm-1} + (m-5).p_{jm-2} + \cdots + (m-5).p_{j3} + (m-3).p_{j2} + (m-1).p_{j1}.$$

KROK 2: Ako suboptimálny rozvrh vezmime rozvrh, pre ktorý je

$$s_{[1]} \geq s_{[2]} \geq \cdots \geq s_{[n]}.$$

Gruptova heuristika pre flow shop $Fm||\mathbf{C_{max}}$.

KROK 1: Pre $j = 1, 2, \ldots, n$ vypočítajme

$$s_j = \frac{e_j}{\min_{1 \leq k \leq m-1}\{p_{jk} + p_{jk+1}\}},$$

kde

$$e_j = 1 \quad \text{ak} \quad p_{j1} < p_{jm} \quad e_j = -1 \quad \text{inak}$$

KROK 2: Ako suboptimálny rozvrh vezmime rozvrh, pre ktorý je

$$s_{[1]} \geq s_{[2]} \geq \cdots \geq s_{[n]}.$$

Uvádza sa, že Gruptova heuristika je vo väčšine prípadov lepšia, než Palmerova.

HEURISTIKA CAMPBEL, DUDEK, SMITH PRE FLOW SHOP $Fm||\mathbf{C_{max}}$.

Táto heuristika má $m-1$ etáp. V každej etape vytvorí z pôvodného $Fm||\mathbf{C_{max}}$ problému dvojstrojový problém $F2||\mathbf{C_{max}}$, v ktorom má úloha J_i časy spracovania p'_{i1}, p'_{i2}. Pre tento problém Johnsonovým algoritmom vypočíta optimálnu permutáciu úloh, a pre túto permutáciu vypočíta v pôvodnom probléme hodnotu $\mathbf{C_{max}}$. Po $m-1$ etapách tak dostaneme $m-1$ permutačných rozvrhov (medzi ktorými môžu byť niektoré i rovnaké), z ktorých vyberieme ten s najmenšou hodnotou $\mathbf{C_{max}}$.
V jednotlivých etapách budú hodnoty p'_{i1}, p'_{i2} nasledovné:

ETAPA 1: $p'_{i1} = p_{i1}$, $p'_{i2} = p_{im}$.

ETAPA 2: $p'_{i1} = p_{i1} + p_{i2}$, $p'_{i2} = p_{im} + p_{im}$.

ETAPA 3: $p'_{i1} = p_{i1} + p_{i2} + p_{i3}$, $p'_{i2} = p_{im-2} + p_{im-1} + p_{im}$.

ETAPA k: $p'_{i1} = \sum_{j=1}^{k} p_{ij}$, $p'_{i2} = \sum_{j=m-k+1}^{m} p_{ij}$.

.

.

.

ETAPA $m-1$: $p'_{i1} = \sum_{j=1}^{k} p_{ij}$, $p'_{i2} = \sum_{j=m-k+1}^{m} p_{ij}$.

Okrem spomínaných heuristík existujú i ďalšie metódy výpočtu optimálneho rozvrhu. Jednou z nich je metóda vetví a hraníc založená na vytváraní čiastočných postupností úloh na jednotlivých strojoch. Pre každú čiastočnú postupnosť treba určiť dolný odhad kriteriálnej funkcie celkového riešenia obsahujúceho túto čiastočnú postupnosť. Ak je dolný odhad daného parciálneho riešnia väčší než doterajší rekord, môžeme toto parciálne riešenie (a s ním i všetky riešenia, ktoré ho obsahujú) zavrhnúť a prejsť k inému parciálnemu riešeniu. Táto metóda je silne závislá na kvalite dolného odhadu – ak ho nemáme dostatočne presný, rastie množstvo parciálnych riešení, ktoré treba preskúmať a časové nároky na výpočet môžu neúnosne rásť. Iným možným prístupom je hľadanie optimálneho rozvrhu pomocou celočíselného resp. bivalentného programovania. V súčasnosti s rastom rýchlosti a pamäťovej kapacity výpočtovej techniky a tiež nebývalým rozvojom metód pre celočíselné lineárne programovanie rastie i význam týchto metód.

Postupy založené na metóde vetví a hraníc resp. na celočíselných modeloch lineárneho programovania sú podobné postupom pre job shop problémy, preto sa na tomto mieste nebudeme nimi podrobnejšie zaoberať.

CHAPTER V

JOB SHOP SYSTÉMY

Je daná množina úloh $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$, množina strojov $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$. Každá úloha J_i bude skladať z m operácií, t.j. $J_i = \{o_{i1}, o_{i2}, \ldots, o_{im}\}$, pričom pre každú úlohu J_i platí

$$o_{i1} \prec\prec o_{i2} \prec\prec \cdots \prec\prec o_{im}.$$

Na rozdiel od flow shop problémov, kde poradie prechodu strojmi bolo rovnaké pre všetky úlohy, v úlohách typu job shop je poradie prechodu rôznych úloh strojmi rôzne – pre každú operáciu o_{ij} ľubovoľnej úlohy J_i je priradený stroj $\mu_{ij} \in \mathcal{M}$, na ktorom sa táto operácia má vykonať s dobou spracovania p_{ij}. V Conwayovej klasifikácii pôjde o systémy $m|n|J|f$, v klasifikácii podľa LLRK $Jm||f$.

Operáciu o_{ij} možno charakterizovať usporiadanou trojicou (i, j, k) vyjadrujúcou skutočnosť, že j–tá operácia i–tej úlohy vyžaduje k–tý stroj M_k. Pretože dvojicou i, j je už jednoznačne určené číslo stroja k, budeme túto skutočnosť vyjadrovať zápisom operácie (i, j, k_{ij}).

Prípustný rozvrh je taký rozvrh, v ktorom sa operácie pre každú úlohu vykonávajú v určenom poradí a kde žiadne dve operácie neobsadzujú ten istý stroj v jednom časovom okamžiku a na žiadnej úlohe sa nevykonávajú dve operácie naraz.

DEFINÍCIA. **Lokálny ľavý posun** (local left shift) v rozvrhu **S** je taký časové posunutie začiatku spracovania niektorej operácie skôr, pri ktorom sa neporuší prípustnosť rozvrhu a zachová sa poradie spracovania operácií na každom stroji.
Globálny ľavý posun (global left shift) je taký presun niektorej operácie do skoršieho časového okamžiku, pri ktorom sa zachová prípustnosť rozvrhu bez zmeny časovej polohy spracovania ostatných operácií.
Semiaktívny rozvrh je taký rozvrh v ktorom neexistuje lokálny **ľavý posun**.
Aktívny rozvrh je taký rozvrh, v ktorom neexistuje globálny ľavý posun.
Zbytočný prestoj je časový interval $\langle t_1, t_2)$, v ktorom sa na niektorom stroji M_j nevykonáva žiadna operácia napriek tomu, že je v systéme operácia, ktorá by sa mohla začať vykonávať v čase t_1.
Rozvrh bez prestojov je taký rozvrh, v ktorom žiaden stroj nestojí v dobe, kedy by mohol vykonávať nejakú operáciu – t.j. rozvrh, v ktorom neexistuje zbytočný prestoj.

Je ľahko vidieť, že ak pre dané poradie úloh pre každý stroj existuje nejaký prípustný rozvrh, potom existuje jediný semiaktívny rozvrh s rovnakým poradím úloh na každom stroji.

Teraz vzniká nasledujúca úloha: Majme dané poradie prechodu úloh pre každý stroj. Treba zostrojiť pre toto poradie úloh semiaktívny rozvrh. Túto úlohu vyriešime nasledujúcim postupom:

Nech $\mathcal{O}$ je množina všetkých operácií. Na $\mathcal{O}$ definujeme reláciu bezprostrednej precedencie $\prec\!\!\prec$ takto:

$$(i,j,k_{ij}) \prec\!\!\prec (i,j+1,k_{ij+1}) \quad \forall i = 1,2\ldots,n \quad \text{a} \quad \forall j = 1,2\ldots,m-1$$

a

$$(i,j,k_{ij}) \prec\!\!\prec (i',j',k_{i'j'}), \quad \text{ak } k_{ij} = k_{i'j'},$$
a úloha J_i bezprostredne predchádza úlohu J_j v poradí pre stroj k_{ij}.

Zostrojme orientovaný graf bezprostrednej precedencie $\mathbb{G}_{\prec\!\!\prec} = (V,H)$, kde $V = \mathcal{O}$, $H = \{[(i,j,k),(i',j',k')]|\ (i,j,k) \prec\!\!\prec (i',j',k')\}$. Ak v grafe $\mathbb{G}_{\prec\!\!\prec}$ existuje cyklus, potom existuje rozpor medzi požiadavkou dodržať poradie spracovania operácií v rámci jednotlivých úloh a požiadavkou na dodržanie poradia úloh na jednotlivých strojoch, a teda v takomto prípade prípustný rozvrh (pre dané poradia na jednotlivých strojoch) neexisatuje. Ak je digraf $\mathbb{G}_{\prec\!\!\prec}$ acyklický, potom postupujeme nasledovne:

Nech $\mathcal{I}$ je množina všetkých nezaradených operácií, nech $\mathcal{I}_0$ je podmnožina $\mathcal{I}$ bez predchodcov. Inicializačne položíme $\mathcal{I} = \mathcal{O}$.

KROK 1: V $\mathcal{I}_0$ neexistujú dve operácie vyžadujúce ten istý stroj, pretože by tieto dve operácie museli byť v relácii $\prec\!\!\prec$, a teda jedna z nich by mala predchodcu. Preto môžeme operácie z $\mathcal{I}_0$ zaraďovať do rozvrhu naraz a zaradíme ich v najskôr možnom čase, ktorý pre každú operáciu (i,j,k) určíme ako maximum dvoch časových okamžikov – ukončenia predchádzajúcej operácie úlohy J_i a okamžiku uvoľnenia stroja M_k.

KROK 2: Položme $\mathcal{I} := \mathcal{I} - \mathcal{I}_0$ a za nové $\mathcal{I}_0$ položme podmnožinu $\mathcal{I}$ bez predchodcov. Ak je $\mathcal{I} = \varnothing$ STOP, inako GOTO KROK 1.

Cvičenie. Flow shop problém je vlastne špeciálnym prípadom job shop problému. Je možné, aby pri pri flow shope nastal prípad, keď v digrafe $\mathbb{G}_{\prec\!\!\prec}$ existuje cyklus?

Tvrdenie. *Množina semiaktívnych rozvrhov je dominantnou množinou pre regulárne optimalizačné kritérium.*

Dôkaz.
Nech $\mathcal{S}$ je ľubovoľný rozvrh, ktorý nie semiaktívny – t.j. existuje v ňom lokálny ľavý posun. Nech $C_1, C_2, \ldots, C_n$ sú časy ukončenia úloh v rozvrhu $\mathcal{S}$. Po realizácii lokálneho ľavého posunu dostaneme nový rozvrh $\mathcal{S}'$ s časmi ukončenia úloh $C'_1, C'_2, \ldots, C'_n$ takými, že $C'_i \leq C_i$ pre $i = 1,2,\ldots,n$. Preto pre regulárne kritérium f platí $f(\mathcal{S}') \leq f(\mathcal{S})$. konečným počtom lokálnych ľavých posunov možno prejsť od ľubovoľného rozvrhu $\mathcal{S}$ k semiaktívnemu rozvrhu $\mathcal{S}''$, ktorého hodnota kriteriálnej funkcie neprevýši hodnotu pôvodného rozvrhu.

TVRDENIE. *Množina aktívnych rozvrhov je dominantnou množinou pre regulárne optimalizačné kritérium.*

Dôkaz.
Je analogický ako dôkaz predchádzajúcej vety.

POZNÁMKA. Nie je záruka, že množina rozvrhov bez prestojov obsahuje optimálny rozvrh.

Úloha hľadať optimálny rozvrh pre job shop systém je vlastne úlohou hľadať prípustné optimálne poradie prechodu úloh jednotlivými strojmi.

Grafový model pre job shop problém.

Zostrojme migraf $\mathbb{G} = (V, H)$, kde $V = \mathcal{O}$, a kde hranová množina H je definovaná nasledovne:

$[(i,j,k),(i',j',k')] \in H$ práve vtedy, keď $i = i'$ a $j' = j + 1$ $((i,j,k),(i',j',k')) \in H$ práve vtedy, keď $k = k'$

Úloha nájsť optimálny rozvrh pre systém $Jm||f$ je ekvivalentná úlohe prideliť neorientovaným hranám migrafu $\mathbb{G}$ také orientácie, aby takto vzniknutý graf $\mathbb{G}$ bol acyklický a aby príslušný rozvrh minimalizoval kriteriálnu funkciu f.

Model celočíselného lineárneho programovania.

Pre účely modelu celočíselného linárneho programovania označme $\mathcal{O}$ množinu všetkých operácií a operácie oindexujme jedným indexom, t.j. $\mathcal{O} = \{o_1, o_2, \ldots, o_N\}$.

Označme:
$\mathcal{O}_0$ množinu indexov všetkých operácií bez predchodcov
$\mathcal{O}_T$ množinu indexov všetkých operácií bez následníkov
x_i čas ukončenia operácie o_i,
p_i dobu spracovania operácie o_i.

Skutočnosť, že rozvrhovanie začína v čase 0, vyjadrujú podmienky:

$$x_i - p_i \geq 0 \quad \forall i \in \mathcal{O}_0 \quad \text{t.j. pre všetky operácie } o_i \text{ bez predchodcov}$$

Označme $I_{\mathcal{J}}$ množinu všetkých usporiadaných dvojíc indexov (i,j) takých, že o_i, o_j sú operácie tej istej úlohy a $o_i \prec\prec o_j$. Ďalej označme $I_{\mathcal{M}}$ množinu všetkých usporiadaných dvojíc indexov (i,j) takých, že $i < j$ a operácie o_i, o_j vyžadujú na spracovanie ten istý stroj. (Podmienka $i < j$ je tu preto, aby $I_{\mathcal{M}}$ obsahovala dvojicu operácií najviac raz).

Bezprostrednú precedenciu operácii v rámci úloh vyjadruje sústava obmedzení:

$$x_j - p_j \geq x_i \quad \forall (i,j) \in I_{\mathcal{J}}$$

Tieto ohraničenia zodpovedajú orientovaným hranám v grafe $\mathbb{G}$. Teraz treba ešte obmedzenia a premenné, ktoré určia orientáciu pre neorientované hrany v $\mathbb{G}$. Pre každú dvojicu $(i,j) \in I_{\mathcal{M}}$ označme y_{ij} binárnu premennú, ktorá je rovná 1 práve vtedy, operácia o_i predchádza operáciu o_j na spoločnom stroji M_k. Pre dve operácie

$o_i,\ o_j$ musí platiť jedna z nerovností

$$\left.\begin{array}{rl} x_j - x_i \geq p_j & \text{ak } y_{ij} = 1 \\ x_i - x_j \geq p_i & \text{ak } y_{ij} = 0 \end{array}\right\} \quad \forall (i,j) \in I_{\mathcal{M}}$$

a teda pre veľmi veľké číslo H musia platiť obe nerovnosti

$$\left.\begin{array}{r} x_j - x_i + H.(1 - y_{ij}) \geq p_j \\ x_i - x_j + H.y_{ij} \geq p_i \end{array}\right\} \qquad \forall (i,j) \in I_{\mathcal{M}} \tag{3}$$

Pretože operácie každej úlohy sú linárne uspriadané, existuje v $\mathcal{O}_T$ pre každú úlohu práve jedna terminálna operácia. Preto množina $\{x_i | i \in \mathcal{O}_T\}$ je množinou časových okamžikov ukončení všetkých úloh z $\mathcal{J}$.

Strednú dobu pobytu úloh v systéme vyjadríme ľahko ako

$$\sum_{i \in \mathcal{O}_T} x_i$$

Nech C predstavuje čas ukončenia poslednej úlohy z dávky $\mathcal{J}$. Potom pre C platí

$$C \geq x_i \qquad \forall i \in \mathcal{O}_T,$$

a pre jeho minimalizáciu stačí zvoliť kriteriálnu funkciu C.

Pre minimalizáciu kritéria $C_{\max}$ máme teda nasledový model:

$$\text{Minimalizovať } C$$

$$\begin{array}{lrll} \text{za predpokladov:} & C \geq x_i & \forall i \in \mathcal{O}_T & \text{(A)} \\ & x_i - p_i \geq 0 & \forall i \in \mathcal{O}_0 & \text{(B)} \\ & x_j - p_j \geq x_i & \forall (i,j) \in I_{\mathcal{J}} & \text{(C)} \end{array}$$

$$\left.\begin{array}{r} x_j - x_i + H.(1 - y_{ij}) \geq p_j \\ x_i - x_j + H.y_{ij} \geq p_i \\ y_{ij} \in \{0,1\} \end{array}\right\} \qquad \forall (i,j) \in I_{\mathcal{M}} \qquad \text{(D)}$$

Pre minimalizáciu kritéria $\sum C_i$ máme tento model:

$$\text{Minimalizovať } \sum_{i \in \mathcal{O}_T} x_i$$

$$\begin{array}{lrll} \text{za predpokladov:} & x_i - p_i \geq 0 & \forall i \in \mathcal{O}_0 & \text{(B)} \\ & x_j - p_j \geq x_i & \forall (i,j) \in I_{\mathcal{J}} & \text{(C)} \end{array}$$

$$\left.\begin{array}{r} x_j - x_i + H.(1 - y_{ij}) \geq p_j \\ x_i - x_j + H.y_{ij} \geq p_i \\ y_{ij} \in \{0,1\} \end{array}\right\} \qquad \forall (i,j) \in I_{\mathcal{M}} \qquad \text{(D)}$$

Greenberg navrhol nasledujúci postup. Najprv sú podmienky (D) vypustené a dostaneme úlohu lineárneho programovania. Ak je výsledné riešenie prípustné, t.j. také, že žiadne dve úlohy nevyžadujú ten istý stroj naraz, máme optimálne riešenie. Ak nastal konflikt pre operácie o_i, o_j), riešime dva subproblémy, jeden s dodatočnou podmienkou

$$x_j - x_i \geq p_j \tag{4,}$$

druhý

$$x_i - x_j \geq p_i \tag{5.}$$

Riešenie každého z nich nám dá dolnú hranicu kriteriálnej funkcie. Ak je niektorá z dolných hraníc väčšia, než doterajší rekord, niet dôvodu pokračovať vo vetvení. Ak je niektoré z riešení prípustné, porovnáme hodnotu jeho kriteriálnej funkcie s doterajším rekordom, a ak došlo k zlepšeniu. aktualizujeme rekord. Takto pokračujeme, kým nepreskúmame všetky perspektívne vetvy stromu riešení.

Algoritmus na generovanie aktívnych rozvrhov (Giffler, Thompson)

Predpokladajme, že máme všetky operácie oindexované jedným indexom. Teda $\mathcal{O} = \{o_1, o_2, \ldots, o_N\}$ nech je množina všetkých operácií, p_i čas spracovania operácie o_i.

Označme

k – počet zaradených operácií

$\mathcal{P}_k$ – parciálny rozvrh obsahujúci k zaradených operácií.

S_k – množina zaraditeľných operácií prislúchajúca parciálnemu rozvrhu $\mathcal{P}_k$

z_i – najskôr možný začiatok operácie $i \in S_k$

k_i – $k_i = y_i + p_i$, t.j. najskôr možný koniec operácie $i \in S_k$

Ak je daný aktívny parciálny rozvrh $\mathcal{P}_k$, potom k nemu prislúchajúca množina zaraditeľných úloh S_k je podmnožinou množiny nezaradených úloh bez nezaradených predchodcov. Pre ľubovoľné $i \in S_k$ určíme z_i ako maximum z času ukončenia jeho priameho predchodcu a z času uvoľnenia stroja, ktorý vyžaduje operácia o_i.

KROK 1: Položme $k := 0$, $\mathcal{P}_k$ nech je prázdny čiastočný rozvrh a S_k obsahuje všetky operácie bez predchodcov.

KROK 2: Určíme $k^* = \min_{i \in S_k}\{k_i\}$. Označme o^* tú operáciu z množiny S_k, ktorá končí v čase k^*. Označme m^* stroj, ktorý vyžaduje operácia o^*. Stroj m^* nazveme strojom asociovaným s k^*.

KROK 3: Pre každú operáciu $i \in S_k$, ktorá vyžaduje stroj m^* a pre ktorú je $z_i < k^*$ vytvorme nový parciálny rozvrh $\mathcal{P}_{k+1}$ ktorý vznikne pridaním operácie i k rozvrhu $\mathcal{P}_k$ tak, že operácia i začne v čase z_i.

KROK 4: Pre každý nový parciálny rozvrh $\mathcal{P}_{k+1}$ vytvorený v kroku 3 aktualizujme dáta nasledovne:
a) Vylúčme operáciu i z množiny S_k, t.j. $S_k := S_k - \{i\}$.
b) Vytvorme S_{k+1} dodaním priameho následníka operácie i k S_k.
b) Inkrementujme k, t.j. $k := k + 1$.

KROK 5: GOTO KROK 2 pre každý čiastočný rozvrh $\mathcal{P}_k$ vytvorený v KROKU 3 a pokračujme týmto spôsobom, pokiaľ nevygenerujeme všetky aktívne rozvrhy.

Kľúčovým miestom vyššie popísaného algoritmu, ktoré zaručuje generovanie aktívnych rozvrhov je krok 3, v ktorom je podmienka $z_i < k^*$ pre zaradenie ďalšej operácie o_i do rozvrhu. Okamžik k^* je totiž časom ukončenia nejakej zaraditeľnej operácie o^* vyžadujúcej stroj m^*. Ak by sme v tomto kroku zaradili do poradia pre stroj m^* operáciu o_i so začiatokm po čase k^*, t.j. takú, že $k^* \leq z_i$, vznikol by tak rozvrh, v ktorom je operácia o_i zaradená pred operáciou o^* a v ktorom by príslušný stroj m^* ostal bez práce po nejaký časový interval $\langle \tau, k^*)$ dostatočne dlhý na presunutie operácie o^* pred operáciu o_i – výsledný rozvrh by nebol aktívny.

V kroku 3 algoritmu teda pre každú zaraditeľnú operáciu o_i takú, že jej spracovanie vyžaduje stroj m^* a $z_i < k^*$ konštruujeme parciálny rozvrh $\mathcal{P}_{k+1}$. Môže sa však stať, že stroj m^* nie je jednoznačne určený (je ich viac), a potom treba krok 3 rozšíriť na každú zaraditeľnú operáciu o_i, $z_i < k^*$ ktorá vyžaduje jeden zo strojov asociovaných s k^*.

Všimnime si ešte, že S_k obsahuje pre každú úlohu prvú nezaradenú operáciu (tak by sa aj dala definovať). S_k môže preto obsahovať najviac n prvkov.

Algoritmus na generovanie non delay rozvrhov

Označme

k – počet zaradených operácií

$\mathcal{P}_k$ – parciálny rozvrh obsahujúci k zaradených operácií.

S_k – množina zaraditeľných operácií prislúchajúca parciálnemu rozvrhu $\mathcal{P}_k$

z_i – najskôr možný začiatok operácie $i \in S_k$

k_i – $k_i = y_i + p_i$, t.j. najskôr možný koniec operácie $i \in S_k$

KROK 1: Položme $k := 0$, $\mathcal{P}_k$ nech je prázdny čiastočný rozvrh a S_k obsahuje všetky operácie bez predchodcov.

KROK 2: Určíme $z^* = \min_{i \in S_k}\{z_i\}$. Označme m^* stroj, ktorý vyžaduje operácia z^*.

KROK 3: Pre každú operáciu $i \in S_k$, ktorá vyžaduje stroj m^* a pre ktorú je $z_i = k^*$ vytvorme nový parciálny rozvrh $\mathcal{P}_{k+1}$ ktroý vznikne pridaním operácie i k rozvrhu $\mathcal{P}_k$ tak, že operácia i začne v čase z_i.

KROK 4: Pre každý nový parciálny rozvrh $\mathcal{P}_{k+1}$ vytvorený v kroku 3 aktualizujme dáta nasledovne:
a) Vylúčme operáciu i z množiny S_k, t.j. $S_k := S_k - \{i\}$.
b) Vytvorme S_{k+1} dodaním priameho následníka operácie i k S_k.
b) Inkrementujme k, t.j. $k := k + 1$.

KROK 5: GOTO KROK 2 pre každý čiastočný rozvrh $\mathcal{P}_k$ vytvorený v KROKU 3 a pokračujme týmto spôsobom, pokiaľ nevygenerujeme všetky nondelay rozvrhy.

Počet nondelay rozvrhov generovaných posledným algoritmom je obyčajne značne menší, než počet aktívnych rozvrhov. Vyplýva to z toho, že pre každý parciálny rozvrh je $z^* < k^*$, a preto počet operácií, pre ktoré je $z_i = z^*$ je menší alebo rovný počtu operácií, pre ktoré je $z_i < k^*$.

Shifting Bottleneck heuristika

Nech $\mathcal{M}$ je množina všetkých strojov. Pri popise jednej iterácie heuristiky budeme predpokladať, že už máme určené orientácie pre disjunktívne hrany prislúchajúce podmnožine $\mathcal{M}_0$ strojov – t.j. pre každý stroj $M \in \mathcal{M}_0$ je už určené poradie prechodu úloh.

Výsledkom iterácie je:

- určenie stroja $M \in \mathcal{M} - \mathcal{M}_0$, ktorý priradíme k množine $\mathcal{M}_0$ a tiež
- určenie poradie spracovanie úloh na tomto stroji.

Pri výbere ďalšieho stroja do množiny $\mathcal{M}_0$ s pokúšame určiť, ktorý nezaradený stroj spôsobuje v nejakom zmysle najväčšie problémy. Na to aby sme to zistili, v pôvodnom digrafe zrušíme všetky disjunktívne hrany prislúchajúce ešte nezaradeným strojom – t.j. strojom z $\mathcal{M} - \mathcal{M}_0$ a ponecháme vybraté disjunktívne hrany prislúchajúce strojm z $\mathcal{M}_0$. Dostaneme tak acyklický graf G' modelujúci situáciu, kedy sa operácie vyžadujúce ten istý stroj, môžu vykonať paralelne, čo je vlastne systém $P\infty|prec|C_{\max}$, ak považujeme každú operáciu za samostatnú úlohu. Je ľahko vypočítať dĺžku príslušného optimálneho rozvrhu $C_{\max}(\mathcal{M}_0)$ a tiež pre každú operáciu o_i časové okno $\langle r_i, d_i)$, v ktorom sa má operáci o_i vykonať tak, aby bola dodržaná dĺžka rozvrhu $C_{\max}(\mathcal{M}_0)$.

Vezmime teraz ľubovoľný stroj $M \in \mathcal{M} - \mathcal{M}_0$ všetky operácie, ktoré sa majú na tomto stroji vykonať. Každá z operácií má určené časové okno, v ktorom sa má vykonať. Pravdepodobne sa nepodarí naplánovať operácie tak, aby skutočne padli do svojich časových okien a pritom ich doby spracovania sa neprekrývali. Riešením problému $1|r_i|L_{\max}$ taký rozvrh pre stroj M, ktorý sa pokiaľ možno najviac približuje daným časovým oknám. Za kritický stroj M vezmeme ten, ktorý ma najväčšiu hodnotu $L_{\max}(M)$.

Platí:

$$C_{\max}(\mathcal{M}_0 \cup \{M\}) = C_{\max}(\mathcal{M}_0) + L_{\max}(M).$$

Položíme

$$\mathcal{M}_0 := \mathcal{M}_0 \cup \{M\}$$

a orientáciu disjunktívnych hrán v grafe problému určíme podľa poradia úloh daného riešením problému $1|r_i|L_{\max}$ pre stroj M.

Takto pokračujem, kým neurčíme orientácie všetkých disjunktívnych hrán.

CHAPTER VI

OPEN SHOP SYSTÉMY

Je daná množina úloh $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$, množina strojov $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$. Každá úloha J_i bude skladať z m operácií, t.j. $J_i = \{o_{i1}, o_{i2}, \ldots, o_{im}\}$. Pre každú operáciu o_{ij} ľubovoľnej úlohy J_i je priradený jediný stroj z množiny $\mathcal{M}$, na ktorom sa má táto operácia vykonávať. Bez ujmy na všeobecnosti možno predpokladať, že j–tá operácia úlohy každej úlohy J_i sa má vykonať na stroji M_j s dobou spracovania p_{ij}. Na rozdiel od flow shop a job shop problémov, kde bolo dané poradie vykonávania operácií v rámci každej úlohy, v problémoch typu open shop je dovolené ľubovoľné poradie vykonávania operácií.

Uvoľnenie precedenčnej relácie nenesie so sebou zjednodušenie problému. Pri job shope sme videli, že k niektorému poradiu prechodu úloh na jednotlivých strojoch vôbec neexistuje prípustný rozvrh. Pri flow shope síce ku každému poradiu prechodu úloh strojmi existuje prípustný rozvrh, ale niektoré poradia generujú vzhľadom na precedenciu operácií v rámci jednotlivých úloh tak zlé rozvrhy, že ich pri riešení použitím metódy branch and bound veľmi skoro možno vylúčiť. Pri open shope ku každému poradiu prechodu úloh strojmi existuje prípustný rozvrh, z ktorých žiadny nevylúči ani príliš nezkazí precedencia operácií v rámci úlohy.

Dosť dobre to vidno na celočíselnom modeli. Pre účely tohot modelu označme $\mathcal{O}$ množinu všetkých operácií a operácie oindexujme jedným indexom, t.j. $\mathcal{O} = \{o_1, o_2, \ldots, o_N\}$. Označme $I_{\mathcal{O}}$, $I_{\mathcal{M}}$ množiny usporadaných dvojíc (i, j) indexov také, že $i < j$ a $(i, j) \in I_{\mathcal{O}}$ práve vtedy, keď operácie o_i, o_j patria k jednej úlohe, práve vtedy, ak operácie o_i, o_j vyžadujú na spracovanie ten istý stroj. Množina $I_{\mathcal{O}} \cup I_{\mathcal{M}}$ je množinou takých dvojíc indexov (i, j), $i < j$, že operácie o_i, o_j nesmú byť spracovávané súčasne.

Ak označíme:
x_i čas ukončenia operácie o_i,
p_i dobu spracovanie operácie o_i,
dostaneme nasledujúci model pre systém $Om||C_{\max}$:

$$\text{Minimalizovať } C$$

$$\text{za predpokladov:} \qquad C \geq x_i \quad \forall i = 1, 2, \ldots, N \tag{A}$$

$$x_i - p_i \geq 0 \quad \forall i = 1, 2, \ldots, N \tag{B}$$

$$\left.\begin{aligned} x_j - x_i + H.(1 - y_{ij}) &\geq p_j \\ x_i - x_j + H.y_{ij} &\geq p_i \\ y_{ij} &\in \{0,1\} \end{aligned}\right\} \qquad \forall (i,j) \in I_{\mathcal{O}} \cup I_{\mathcal{M}} \tag{D}$$

Ak porovnáme tento model s modelom pre job shop vidíme, že tu je podstatne viac celočíslených premenných y_{ij} a úplne absentujú precedenčné podmienky, čo pravdepodobne zväčší počet prípustných riešení nad možnosti súčasnej výpočtovej techniky.

Problém $O2||C_{\max}$

Majme dva stroje, t.j. $\mathcal{M} = \{M_1, M_2\}$ a n–prvkovú množinu úloh $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$, každá úloha J_i pozostáva z dvoch operácií, $J_i = \{o_{i1}, o_{i2}\}$, pričom operácie o_{i1}, o_{i2} sa vykonajú na strojoch M_1, M_2 s časmi spracovania p_{i1}, p_{i2}. Na poradí spracovania operácií nezáleží. Nájsť dolnú hranicu pre $C_{\max}$ nie je ťažké:

$$C_{\max} \geq \max\left(\sum_{i=1}^{n} p_{i1}, \sum_{i=1}^{m} p_{i2}\right)$$

Algoritmus LAPT pre $O2||C_{\max}$. Kedykoľvek sa uvoľní stroj, zaraď tú operáciu čakajúcu na spracovanie, ktorá má najdlhší čas spracovania na inom stroji.

Veta. *LAPT algoritmus dáva optimálny rozvrh pre* $O2||C_{\max}$*, pričom platí:*

$$C_{\max} = \max\left(\max_{i=1,2\ldots,n}(p_{i1} + p_{i2}), \sum_{i=1}^{n} p_{i1}, \sum_{i=1}^{m} p_{i2}\right)$$

Veta. *Problém* $O3||C_{\max}$ *je NP–ťažký.*

Problém $Om|pmtn|C_{\max}$ je polynomiálne riešiteľný a algoritmus na jeho riešenie je uvedený ako súčasť algoritmu na riešenie problému $Rm|pmtn|C_{\max}$. Pre $Om||C_{\max}$ dáva riešenie príslušného problému s povolením prerušenia operácií dolnú hranicu, ktorá by sa mohla využiť pri riešení pomocou metódy vetví a hraníc.

Všeobecne je open shopom venovaná oveľa menšia pozornosť, ako ostatným systémom so špecializovanými strojmi a daným poradím spracovania operácií.