Stanislav Palúch, Štefan Peško

KVANTITATÍVNE METÓDY V LOGISTIKE

VYDALA ŽILINSKÁ UNIVERZITA V ŽILINE
2006

Táto monografia vznikla vďaka grantovej podpore grantu VEGA 1/0490/03.

Vedecký redaktor: prof. Ing. Mikuláš Alexík, CSc.

Recenzenti: prof. RNDr. Ján Černý, DrSc., Dr.h.c.
doc. RNDr. Bohdan Linda, CSc.

Vydala Žilinská univerzita v Žiline – EDIS vydavateľstvo ŽU

© S. Palúch, Š. Peško, 2006
ISBN-80-8070-636-0

Úvod

Existujú dva prístupy k chápaniu vedného odboru logistika:

- anglosaský prístup – logistika v užšom zmysle, kedy sa do logistických procesov zahŕňajú len procesy obstarávania a distribúcie
- kontinentálny prístup – logistika v širšom zmysle – rozširuje chápanie logistických procesov aj o výrobnú oblasť

Európska logistická asociácia definuje logistiku ako „*Organizovanie, plánovanie, riadenie a výkon tokov začínajúc vývojom a nákupom, končiac výrobou a distribúciou podľa objednávky finálneho zákazníka tak, aby boli splnené všetky požiadavky trhu pri minimálnych nákladoch a minimálnych kapitálových výdavkoch.*“

Minimalizácia nákladov a minimalizácia kapitálových výdavkov má výrazný kvantitatívny charakter, kde sa priam núka použitie optimalizačných metód operačnej analýzy, pokiaľ existujú praktické možnosti na dostatočne presný kvantitatívny popis príslušných logistických procesov.

V zmysle definície Európskej logistickej asociácie možno rozčleniť logistiku na

- výrobnú logistiku
- distribučnú a dopravnú logistiku a
- skladovú a obstarávaciu logistiku.

Ako dlhoroční pracovníci Výskumného ústavu dopravného v Žiline a neskôr Žilinskej univerzity sme sa dlhodobo zaoberali riešením kvantitatívnych problémov všetkých vymenovaných oblastí logistiky. Za ten čas sme zozbierali, vyvinuli, implementovali, overili a v praxi aplikovali množstvo prístupov, metód a algoritmov pre riešenie kvantitatívnych problémov v logistike.

V rámci grantovej úlohy VEGA 1/0490/03 sme dostali priestor na zozbieranie, utriedenie, klasifikáciu, ďalší rozvoj a vytvorenie uceleného systému kvantitatívnych metód v logistike, ktorý teraz v tejto publikácii predkladáme.

Kvantitatívne metódy v tejto publikácii sú rozdelené podľa spomínaného členenia do troch rovnomenných kapitol. Pri spracovaní látky sme zvolili prístup s dôrazom na tvorbu modelov riešených problémov so snahou vytvoriť takú formuláciu väčšiny problémov, ktorá umožní použiť štandardné metódy operačnej analýzy, ako je lineárne programovanie, klasické algoritmy teórie grafov, či približné iteračné metódy, z ktorých mnohé sú súčasťou bežne dostupných programov, ako sú napríklad tabuľkové procesory.

Pre tie problémy, pre ktoré nie je možné použiť štandardné algoritmy, uvádzame exaktné alebo suboptimálne algoritmy, z ktorých sme väčšinu odskúšali a ktoré povedú podľa našich skúseností k dobrým praktickým výsledkom. Pritom sme sa snažili vyberať algoritmy, ktoré sú čo najjednoduchšie implementovateľné.

Ďakujeme grantovej agentúre VEGA a grantu č. 1/0490/03, ktorý nám umožnil vypracovať túto publikáciu.

Ďakujeme tiež prof. RNDr. Jaroslavovi Janáčkovi, CSc. a recenzentom prof. RNDr. Jánovi Černému, DrSc. a doc. Bohdanovi Lindovi, CSc. za mnohé cenné rady, pripomienky odborného i formálneho charakteru. Obzvlášť ďakujeme Ing. Petrovi Palúchovi, ktorý celý rukopis pozorne prečítal a opravil množstvo vecných, formálnych, pravopisných, štylistických a iných chýb.

V Žiline 30.11.2006 Autori

Obsah

Kapitola 1

Prerekvizity

1.1 Základné pojmy z teórie grafov

Túto časť uvádzame preto, lebo terminológia teórie grafov nie je ustálená. Niektorí autori dávajú prednosť takej terminológii, v ktorej sa pod pojmom graf rozumie najvšeobecnejšia štruktúra a ostatné štruktúry ako digraf, neorientovaný graf, atď. sa chápu ako špeciálne prípady grafu. Autori tejto publikácie sú však silne ovplyvnení vynikajúcou knihou [43] J. Plesníka a preto jeho terminológiu prijali za svoju.

1.1.1 Digraf a graf

Digrafom nazveme usporiadanú dvojicu $G = (V, H)$, kde V je neprázdna konečná množina vrcholov digrafu a H je množina usporiadaných dvojíc typu (u, v), kde $u \neq v$, t. j.

$$H \subseteq \{(u, v) \mid u \neq v,\ u, v \in V\} \subset V \times V,$$

kde symbolom $V \times V$ značíme karteziánsky súčin množín V a V – t. j. množinu všetkých usporiadaných dvojíc prvkov z V.
Prvky množiny H nazývame *orientovanými hranami* digrafu G.

Grafom nazveme usporiadanú dvojicu $G = (V, H)$, kde V je neprázdna konečná množina vrcholov grafu a H je množina neusporiadaných dvojíc typu $\{u, v\}$, kde $u \neq v$, t. j.

$$H \subseteq \{\{u, v\} \mid u \neq v,\ u, v \in V\} \subset V \circ V,$$

kde symbolom $V \circ V$ značíme množinu všetkých neusporiadaných dvojíc prvkov z množiny V.
Prvky množiny H nazývame *hranami* grafu G.

Všimnime si, že definície grafu a digrafu neuvažujú tzv. *slučky* – t. j. hrany typu $\{u, u\}$, resp. (u, u), čo však nie je vážny nedostatok, pretože v praxi je úsek spájajúci uzol so sebou samým ojedinelým javom.

Na modelovanie dopravnej siete s číselnými charakteristikami hrán, resp. uzlov zavedieme pojmy hranovo ohodnoteného, resp. vrcholovo ohodnoteného grafu alebo digrafu.

Graf (resp. digraf) $G = (V, H)$ nazveme *hranovo ohodnoteným*, ak každej hrane (resp. orientovanej hrane) $h \in H$ je priradené reálne číslo $c(h)$. Za hranovo ohodnotený graf (resp. digraf) budeme teda pokladať usporiadanú trojicu $G = (V, H, c)$, kde V je množina vrcholov, H množina hrán (resp. orientovaných hrán) a $c : H \to \mathbb{R}$ je reálna funkcia definovaná na množine H.

Podobne možno definovať *vrcholovo ohodnotený graf (digraf)* ako usporiadanú trojicu $G = (V, H, w)$, kde V je množina vrcholov, H množina hrán (resp. orientovaných hrán) a $w : V \to \mathbb{R}$ je reálna funkcia definovaná na množine V.

Graf, resp. digraf sú vhodné na modelovanie takých situácií, kedy z uzla u do iného uzla v vedie nanajvýš jedna hrana $\{u, v\}$, resp. medzi uzlami u a v existuje nanajvýš jedna orientovaná hrana (u, v) a jedna opačne orientovaná hrana (v, u). V týchto prípadoch je totiž hrana jednoznačne určená svojimi koncovými vrcholmi.

Pre prípady so zakázanými slučkami, kedy medzi vrcholmi u, v existuje viac hrán máme *multigraf*, resp. *multidigraf*. Pre modelovanie situácií so slučkami (napríklad slučka na konečnej stanici električky) máme *pseudograf*, resp. pseudodigraf, v ktorých sú okrem viacnásobných hrán dovolené i slučky.

Najvšeobecnejšou grafovou štruktúrou je *pseudomigraf* obsahujúci orientované i neorientované viacnásobné hrany a slučky. Presné matematické definície všetkých spomínaných štruktúr sú už zložitejšie a čitateľ ich môže nájsť v [43]. V tejto publikácii sa vyskytnú iba ojedinele a vtedy ich budeme chápať intuitívne.

1.1.2 Cesty v digrafoch a grafoch

V digrafoch, resp. grafoch môžeme definovať niekoľko spôsobov cestovania po grafe. Sú to orientovaný sled, ťah, cesta a cyklus.
Orientovaný sled z vrchola v_1 do vrchola v_n alebo tiež v_1–v_n sled v digrafe G je ľubovoľná alternujúca (striedavá) postupnosť vrcholov a orientovaných hrán

tvaru

$$m(v_1, v_n) = v_1, (v_1, v_2), v_2, (v_2, v_3), v_3, \ldots, (v_{n-1}, v_n), v_n, \tag{1.1}$$

kde $v_i \in V$ pre $i = 1, 2, \ldots, n$ a $(v_i, v_{i+1}) \in H$ pre $i = 1, 2, \ldots, n-1$. Vrchol v_1 je *počiatočný vrchol*, vrchol v_n *koncový vrchol* sledu (1.1). Sled (1.1) nazveme *uzavretý*, ak $v_1 = v_n$.
Orientovaný ťah v digrafe G je taký orientovaný sled v digrafe G, v ktorom sa žiadna hrana neopakuje.
Orientovaná cesta v digrafe G je taký orientovaný sled v digrafe G, v ktorom sa žiaden vrchol neopakuje. *Orientovaný cyklus* je uzavretý orientovaný ťah, v ktorom sa okrem prvého a posledného vrchola žiaden vrchol nevyskytuje viac než raz.
Dĺžkou orientovaného sledu $m(u, v)$ nazveme súčet ohodnotení jeho hrán, pričom ohodnotenie každej hrany započítavame toľkokrát, koľkokrát sa táto hrana v slede vyskytuje. Dĺžku sledu $m(u, v)$ budeme značiť $d(m(u, v))$.

Pre grafy definujeme analogicky *sled*, *ťah*, *cestu* a *cyklus*.

Ak v digrafe nezáleží na orientácii hrán – t. j. ak hrana môže byť v slede použitá aj v protismere, dostaneme *polosled*, *poloťah*, *polocestu* a *polocyklus*.

Presnejšie polosled definujeme ako alternujúcu postupnosť vrcholov a orientovaných hrán digrafu tvaru

$$m(v_1, v_n) = v_1, h_1, v_2, h_2, \ldots, v_{n-1}, h_{n-1}, v_n,$$

kde $h_i = (v_i, v_{i+1})$, v tom prípade hovoríme, že orientovaná hrana h_i je *použitá v poloslede v smere orientácie*, alebo $h_i = (v_{i+1}, v_i)$, v tom prípade hovoríme, že orientovaná hrana h_i je *použitá v poloslede proti smeru orientácie*.

V grafoch a digrafoch stačí na určenie sledu aj postupnosť vrcholov z (1.1). Uvedením hrán však dostávame definíciu vhodnú aj pre multigrafy. Naviac má význam hovoriť, že hrana $\{i, j\}$ patrí, resp. nepatrí do sledu $m(u, v)$.

Ak graf, resp. digraf G neobsahuje cyklus, resp. orientovaný cyklus, hovoríme, že G je *acyklický* graf, resp. digraf. . Hovoríme, že graf G je *súvislý*, , ak pre ľubovoľné dva jeho vrcholy u, v existuje u–v cesta. *Strom* je súvislý acyklický digraf.

1.2 Algoritmy a ich zložitosť

1.2.1 Algoritmy

Pod pojmom *algoritmus* rozumieme postupnosť krokov, resp. pravidiel, ktorá nás dovedie k žiadanému riešeniu daného problému. Žiada sa, aby algoritmus mal tieto vlastnosti:

- determinovanosť – má byť zadaný konečným počtom jednoznačných pravidiel
- efektívnosť – má zaručiť vyriešenie úlohy po konečnom počte krokov
- hromadnosť – má byť použiteľný na celú triedu prípadov úlohy svojho typu

Pravidlá v algoritme musia byť rozhodnuteľné v okamihu výpočtu. Príklad v súčasnosti nerozhodnuteľného pravidla je nasledujúci: „Ak existuje párne číslo väčšie než 2, ktoré sa nedá vyjadriť ako súčet dvoch prvočísel [1], choď na krok A, inak skoč na krok B.“ Pretože nevieme, či možno všetky párne čísla väčšie než 2 vyjadriť ako súčet dvoch prvočísel, nevieme, ako ďalej vo výpočte pokračovať.

Jedným z najznámejších algoritmov je *Euklidov algoritmus* na hľadanie najväčšieho spoločného deliteľa NSD(a, b) dvoch prirodzených čísel $a,\ b$.

Algoritmus možno zapísať napríklad postupnosťou krokov, blokovou schémou, príkazmi programovacieho jazyka a iste mnohými ďalšími spôsobmi. Existuje však aj niekoľko presných matematických formalizácií pojmu algoritmu – Turingove počítače, Markovove algoritmy a RAM počítače. Tieto formalizácie umožnili dokázať, že neexistuje algoritmus, ktorý by dokazoval všetky vety elementárnej aritmetiky (Gödel 1931). Postupom času sa našla celá trieda algoritmicky nerozhodnuteľných problémov. Na účely tejto publikácie však vystačíme s intuitívnym chápaním pojmu algoritmus.

1.2.2 Zložitosť algoritmov a problémov

Neodmysliteľnou súčasťou moderného prístupu k štúdiu problémov a algoritmov je rozbor obťažnosti riešeného problému a stanovenie výpočtovej zložitosti navrhnutého algoritmu. Pojmy ako „*obťažnosť*“, „*výpočtová zložitosť*“ treba najprv

[1] Ide o tzv. Goldbachovu domnienku z roku 1742, že každé párne číslo väčšie než 2 možno vyjadriť ako súčet dvoch prvočísel. Goldbachova domnienka patrí medzi najstaršie nevyriešené problémy matematiky.

korektne matematicky definovať a potom na základe týchto definícií ukázať vlastnosti skúmaných problémov. Týmito záležitosťami sa zaoberá teória výpočtovej zložitosti, ktorá dnes dospela do značnej dokonalosti. Zistila, že existujú „ľahké“ a „ťažké“ problémy a zostavila celú hierarchiu obťažnosti problémov. Vďaka nej máme dnes celé zoznamy „ťažkých“ problémov.

Na naše účely však bude podstatné rozoznať základný rozdiel medzi „ľahkými“ a „ťažkými“ problémami. Preto cieľom tejto kapitoly je podať základy výpočtovej zložitosti algoritmov v najjednoduchšej možnej forme.

Ukázalo sa rozumné posudzovať algoritmy podľa počtu elementárnych krokov, ktoré potrebuje na vyriešenie daného problému. Tieto elementárne kroky môžu byť sčítanie, odčítanie, násobenie, delenie, porovnávanie s vetvením atď. Jednotlivé elementárne kroky považujeme za rovnako časovo náročné.

Definícia 1.1. Budeme hovoriť, že *algoritmus vyrieši danú konkrétnu úlohu U v čase T*, ak na jej vyriešenie potrebuje T elementárnych krokov.

Je evidentné, že čas T závisí od množstva údajov úlohy U. Tak napríklad usporiadanie postupnosti čísel podľa veľkosti (tzv. triedenie) bude závisieť od počtu členov tejto postupnosti. Celkový počet dát potrebných na zadanie úlohy U budeme volať *dĺžka úlohy U*.

Počet elementárnych krokov T potrebných na vyriešenie úlohy U bude závisieť aj od vnútornej štruktúry úlohy. Tak napríklad niektoré triediace algoritmy skončia skôr, keď dostanú ako vstup utriedenú postupnosť. Z uvedených dôvodov sa presné počty elementárnych operácií ťažko získavajú, a preto sa uspokojujeme s odhadmi času T pre veľké dĺžky úloh n.

Definícia 1.2. Nech $g(n)$, $h(n)$ sú dve kladné funkcie definované na množine prirodzených čísiel. Budeme písať $g(n) = O(h(n))$ a $h(n) = \Omega(g(n))$ a hovoriť, že *funkcia $h(n)$ asymptoticky dominuje funkcii $g(n)$*, ak existuje konštanta K a prirodzené číslo n_0 také, že

$$g(n) \leq Kh(n) \quad \text{pre každé } n \geq n_0.$$

Hovoríme, že *algoritmus $\mathcal{A}$ má zložitosť $O(f(n))$*, resp. *A je $O(f(n))$ algoritmus*, ak pre každé dostatočne veľké n vyrieši úlohu dĺžky n v čase $O(f(n))$. Analogicky hovoríme, že algoritmus $\mathcal{A}$ *má zložitosť $\Omega(f(n))$*, resp. *A je $\Omega(f(n))$ algoritmus*, ak pre každé n_0 existuje príklad úlohy dĺžky $n \geq n_0$, na vyriešenie ktorého potrebuje čas $\Omega(f(n))$.

Špeciálne ak $\mathcal{A}$ je $O(f(n))$ algoritmus, kde za $f(n)$ môžeme zobrať polynóm, hovoríme, že $\mathcal{A}$ je *polynomiálny algoritmus*.

Hovoríme, že *problém má zložitosť* $O(f(n))$, ak preň existuje $O(f(n))$ algoritmus. Problém má *zložitosť* $\Omega(f(n))$, ak každý algoritmus preň je $\Omega(f(n))$ algoritmus.

1.2.3 Úloha lineárneho programovania

Definícia 1.3. *Úloha lineárneho programovania* je nájsť také reálne čísla $x_1, x_2, \ldots, x_n$, pre ktoré je

$$f(\mathbf{x}) = \mathbf{c}^T.\mathbf{x} = c_1x_1 + c_2x_2 + \cdots + c_nx_n \quad \text{minimálne}$$

za predpokladov:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots & \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m \\ \\ x_1, x_2, \ldots, x_n &\geq 0 \end{aligned}$$

Skrátene pomocou maticových zápisov možno úlohu lineárneho programovania formulovať ako: Minimalizovať $\mathbf{c}^T.\mathbf{x}$ za predpokladov $\mathbf{A}.\mathbf{x} = \mathbf{b}$, $\mathbf{x} \geq \mathbf{0}$. Na riešenie úlohy lineárneho programovania máme napríklad slávnu simplexovú metódu, ktorou (alebo jej modifikáciami) zvládneme s dnešnou výpočtovou technikou aj úlohy s tisíckami premenných.

Ďalšie praktické požiadavky môžu požadovať, aby premenné $x_1, x_2, \ldots, x_n$ predstavovali počty reálnych objektov. V tomto prípade musia byť všetky premenné $x_1, x_2, \ldots, x_n$ celé čísla. Špeciálnym prípadom je prípad, keď požadujeme, aby všetky premenné nadobúdali len hodnoty 0 alebo 1.

Definícia 1.4. *Úloha celočíselného lineárneho programovania* (úloha CLP) je nájsť takú n–ticu celých čísiel $\mathbf{x}$, pre ktorú je $\mathbf{c}^T.\mathbf{x}$ minimálne za predpokladov $\mathbf{A}.\mathbf{x} = \mathbf{b}$, $\mathbf{x} \geq \mathbf{0}$.
Úloha bivalentného (alebo binárneho) lineárneho programovania (úloha BLP) je nájsť takú n–ticu celých čísiel $\mathbf{x}$, pre ktorú je $\mathbf{c}^T.\mathbf{x}$ minimálne za predpokladov $\mathbf{A}.\mathbf{x} = \mathbf{b}$, $x_i \in \{0, 1\}$ pre $i = 1, 2, \ldots, n$.

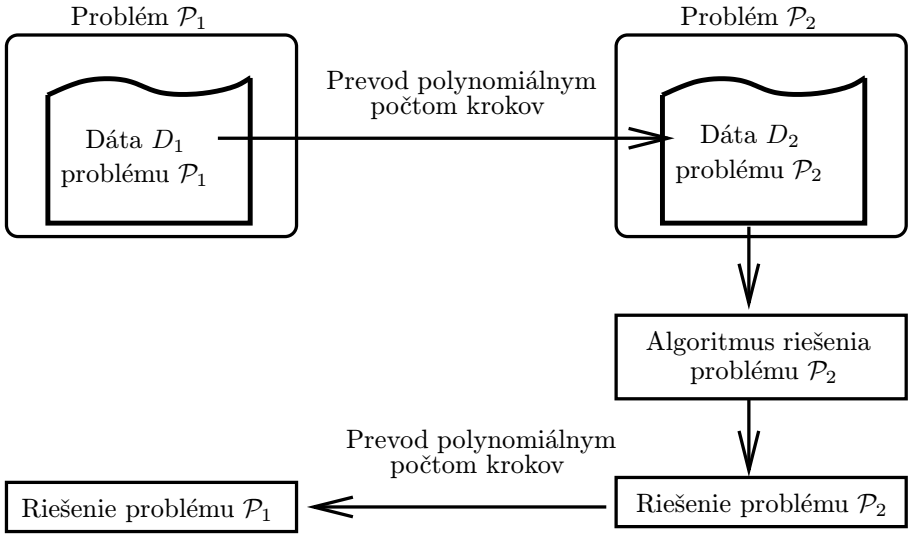
Úloha celočíselného lineárneho programovania je už podstatne ťažšia ako úloha lineárneho programovania. Úloha bivalentného programovania je špeciálnym prípadom úlohy celočíselného lineárneho programovania, avšak rovnako patrí medzi najťažšie úlohy kombinatorickej matematiky. Existujú metódy a programy na riešenie oboch typov úloh, rozsah riešených príkladov je

však podstatne menší než príkladov lineárneho programovania, pri ktorom sa nežiada celočíselnosť, resp. bivalentnosť riešenia.

1.2.4 Polynomiálne transformácie

Častým postupom pri riešení nejakého problému je previesť ho na jedno alebo viac riešení iného problému. Tak napríklad násobenie dvoch celých čísiel vieme previesť na sériu sčítaní. Hľadanie najväčšieho spoločného deliteľa dvoch čísiel prevedieme na postupnosť delení so zvyškom.

Nech je daný problém $\mathcal{P}_1$ so vstupnými dátami D_1. Problém $\mathcal{P}_1$ môžeme riešiť aj tak, že ho pretransformujeme na iný problém $\mathcal{P}_2$ tak, že dáta D_1 prevedieme na dáta D_2 problému $\mathcal{P}_2$. Ak riešenie R_2 problému $\mathcal{P}_2$ vieme previesť na riešenie R_1 problému $\mathcal{P}_1$, transformácia je hotová. Ak prevod dát D_1 na D_2 a riešenia R_2 na R_1 vyžaduje len polynomiálny počet elementárnych operácií, nazveme túto transformáciu *polynomiálnou transformáciou*.



Obr. 1.1: Polynomiálna transformácia problému $\mathcal{P}_1$ na problém $\mathcal{P}_2$

Ak problém $\mathcal{P}_1$ možno polynomiálne transformovať na problém $\mathcal{P}_2$ a problém $\mathcal{P}_2$ má polynomiálny algoritmus riešenia, potom aj problém $\mathcal{P}_1$ má polynomiálny algoritmus riešenia, pretože prevod dát D_1 na dáta D_2, získanie riešenia R_2

problému $\mathcal{P}_2$ s dátami D_2 a prevod riešenia R_2 na riešenie R_1 problému $\mathcal{P}_1$ možno urobiť polynomiálnym počtom krokov.

V niektorých prípadoch vyriešenie problému $\mathcal{P}_1$ vyžaduje vyriešiť niekoľko prípadov problému $\mathcal{P}_2$ (napr. násobenie prevádzame na niekoľko sčítaní). Ak prevody dát a riešení vyžadujú len polynomiálny počet elementárnych operácií a výpočet vyžaduje polynomiálny počet výpočtov problému $\mathcal{P}_2$, hovoríme, že sme problém $\mathcal{P}_1$ *polynomiálne redukovali na problém* $\mathcal{P}_2$. Ak problém $\mathcal{P}_1$ možno polynomiálne redukovať na problém $\mathcal{P}_2$ a naopak, problém $\mathcal{P}_2$ možno polynomiálne redukovať na $\mathcal{P}_1$ hovoríme, že problémy $\mathcal{P}_1$, $\mathcal{P}_2$ sú *polynomiálne ekvivalentné*.

1.2.5 NP–ťažké úlohy a NP–ekvivalentné úlohy

Mnohé optimalizačné úlohy kvantitatívnej logistiky možno zaradiť k úlohám diskrétnej optimalizácie. Riešenie takýchto úloh spočíva vo výbere jedného z konečného počtu kombinatorických objektov. Najjednoduchším spôsobom riešenia takýchto úloh je úplné prehľadanie všetkých objektov (full search, brute force). Klasická matematika takéto problémy skutočne rieši takto – prehľadaním konečného počtu prvkov nájdeme ten hľadaný.

Rozsah úplného prehľadávania však môže byť obrovský – počet všetkých podmnožín n-prvkovej množiny je 2^n, počet permutácií n-prvkovej množiny je $n!$, počet všetkých zobrazení n-prvkovej množiny do seba je n^n. Všetky uvedené funkcie rastú oveľa rýchlejšie než akýkoľvek polynóm premennej n.

Edmons (1965) bol prvý, kto si uvedomil rozdiel medzi polynomiálnymi algoritmami (t. j. algoritmami so zložitosťou typu $O(n^k)$) a nepolynomiálnymi algoritmami, ktorých počet krokov nevieme ohraničiť polynómom. Tie prvé nazýva „dobré“. Podľa neho aj problémy možno rozdeliť na také, pre ktoré existuje polynomiálny algoritmus a tie ostatné – „beznádejné“, pre ktoré takýto algoritmus neexistuje. Túto ideu sa zatiaľ nepodarilo plne realizovať, pretože neexistenciu polynomiálneho algoritmu sa pre väčšinu problémov považovaných za „beznádejné“ nepodarilo dokázať.

Ak úlohu $\mathcal{P}_1$ možno polynomiálne redukovať na úlohu $\mathcal{P}_2$, znamená to toto: Ak existuje polynomiálny algoritmus riešenia úlohy $\mathcal{P}_2$, potom existuje aj polynomiálny algoritmus pre úlohu $\mathcal{P}_1$. Naopak to však nemusí platiť – ak existuje polynomiálny algoritmus pre $\mathcal{P}_1$, polynomiálna redukovateľnosť $\mathcal{P}_1$ na $\mathcal{P}_2$ ešte nič nehovorí o zložitosti problému $\mathcal{P}_2$.

Ak sú však problémy $\mathcal{P}_1$, $\mathcal{P}_2$ polynomiálne ekvivalentné, existencia polynomiálneho algoritmu pre jeden implikuje existenciu polynomiálneho algoritmu pre druhý.

Ako etalón ťažkých úloh bola vybratá úloha binárneho lineárneho programovania (BLP). Problém, ktorý možno polynomiálne redukovať na úlohu BLP, je ľahší ako úloha BLP. Problém, na ktorý možno polynomiálne redukovať úlohu BLP je ťažší ako úloha BLP.

Definícia 1.5. Hovoríme, že *problém $\mathcal{P}$ je NP–ťažký*, ak úlohu binárneho lineárneho programovania možno polynomiálne redukovať na $\mathcal{P}$.
Hovoríme, že *problém $\mathcal{P}$ je NP–ľahký*, ak problém $\mathcal{P}$ možno polynomiálne redukovať na úlohu binárneho lineárneho programovania.
Hovoríme, že *problém $\mathcal{P}$ je NP–ekvivalentný*, ak je problém $\mathcal{P}$ polynomiálne ekvivalentný s úlohou bivalentného lineárneho programovania.

Doteraz sa podarilo pre stovky úloh dokázať, že sú NP–ekvivalentné. Pre ďalšie sa podarilo dokázať, že sú NP-ťažké. Pre žiadnu NP–ťažkú úlohu sa doteraz nepodarilo nájsť polynomiálny algoritmus. Nepodarilo sa však ani dokázať, že polynomiálny algoritmus pre ne neexistuje. Nájdenie takého algoritmu pre jedinú zo stoviek NP–ekvivalentných úloh by znamenalo existenciu polynomiálneho algoritmu pre všetky ostatné. Všeobecne sa neverí, že by sa to mohlo niekomu podariť, preto sú na NP–zložitosti niektorých úloh založené napr. aj niektoré kryptografické systémy.

Kapitola 2

Výrobná logistika

2.1 Základné pojmy teórie výrobných rozvrhov

Vo svetovej literatúre existuje niekoľko prístupov k teórii rozvrhovania. Všeobecná teória rozvrhovania zahŕňa okrem rozvrhov výrobných aj zložitejšie rozvrhy ako sú rozvrhy školské, dopravné, osobné a mnohé iné. Všeobecnému prístupu k rozvrhovacej problematike je potom podriadená aj terminológia, v ktorej sa používajú pojmy *actor*, *transformation*, resp. *object* namiesto pojmov *machine*, *operation*, resp. *job*. Keďže sa v tejto kapitole obmedzíme na výrobné rozvrhy, prikloníme sa k terminológii výrobného rozvrhovania, ako je používaná v [5], [42].

2.1.1 Operácia, úloha, stroj

Základnými pojmami teórie výrobných rozvrhov sú *stroj, operácia* a *úloha*.

Definícia 2.1. *Operácia o* (anglicky *operation*) je základný technologický úkon, ktorý už ďalej nie je deliteľný na čiastočné úkony.

Aj keď je operácia nedeliteľná, niektoré modely pripúšťajú prerušenie operácie a po nejakom čase pokračovanie v jej vykonávaní. Vo väčšine modelov sa pri ďalšom pokračovaní prerušenej operácie pokračuje tam, kde bola daná operácia prerušená.

Definícia 2.2. *Úloha J* (anglicky *job*) je množina operácií $\{o_1, o_2 \ldots, o_m\}$, z ktorých žiadne dve sa nesmú vykonávať v tom istom čase. (Toto obmedze-

nie vyplýva zrejme z toho, že úlohou modelujeme množinu operácií s jedným objektom.)

Definícia 2.3. *Stroj* M je zariadenie schopné vykonať jeden alebo niekoľko druhov operácií. Anglický termín pre stroj je *machine*, v českej terminológii sa používa tiež termín *procesor*.

Pri rozvrhovacom probléme treba pre každú operáciu o_{ij} každej úlohy J_i z danej množiny úloh $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ určiť stroj M_j z danej množiny strojov $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$, na ktorom sa operácia o_{ij} vykoná, a časový interval jej vykonávania (resp. niekoľko časových intervalov v prípade, keď je dovolené prerušenie operácií).

Vo väčšine rozvrhovacích modelov platia dva základné predpoklady:

1. Každý stroj v jednom časovom okamihu môže vykonávať nanajvýš jednu operáciu.
2. V jednom časovom okamihu sa nemôžu vykonávať dve alebo viac operácií jednej úlohy.

Stroje z množiny $\mathcal{M}$ môžu byť univerzálne alebo špecializované. *Univerzálne* stroje môžu spracovávať ľubovoľnú operáciu ľubovoľnej úlohy. Jednotlivé stroje z množiny $\mathcal{M}$ sa môžu líšiť dobou spracovania operácií. Súčasťou riešenia rozvrhovacieho problému je v takom prípade i priradenie strojov jednotlivým operáciám.

Špecializované stroje sú určené len na spracovanie niektorých operácií – každá operácia už má priradený stroj, na ktorom sa bude spracovávať.

Majme úlohu $J = \{o_1, o_2, \ldots, o_k\}$. Pri niektorých úlohách nezáleží na poradí vykonávania týchto operácií. Pri iných z technológie vyplýva, že niektoré operácie možno robiť až po vykonaní iných (napr. najprv je nutné postaviť základy domu, potom je možné stavať múry a až potom strechu). Túto skutočnosť modelujeme tak, že v rámci jednotlivej úlohy alebo i celého rozvrhovacieho problému môže byť daná *precedenčná relácia* $\prec$ medzi jednotlivými operáciami. Ak operáciu y možno začať vykonávať až po skončení operácie x, budeme hovoriť, že operácia x predchádza operáciu y a písať $x \prec y$. Relácia $\prec$ je tranzitívna, t. j.

$$\text{ak} \quad x \prec y \quad \text{a} \quad y \prec z \quad \text{potom} \quad x \prec z$$

Budeme hovoriť, že *operácia x bezprostredne predchádza operáciu y*, ak $x \prec y$ a neexistuje operácia z taká, že $x \prec z \prec y$. Túto skutočnosť budeme označovať ako $x \prec\prec y$.

Často býva vhodné vyjadriť si následnosť operácií jednej úlohy vo forme digrafu $G_{\prec} = (V, H)$, ktorého množinou vrcholov je množina všetkých operácií danej úlohy a množinou orientovaných hrán je $H = \{(x, y) \mid x \prec y\}$. Digraf $G_{\prec}$ nazveme *precedenčným digrafom* alebo *digrafom precedencie* $\prec$ na príslušnej množine operácií. (Iným – podobným, ale nie totožným – spôsobom je vytvorenie sieťového digrafu danej úlohy, tam sa však operácie – elementárne činnosti modelujú hranami).

Analogicky môže byť definovaná relácia precedencie $\prec$ i bezprostrednej precedencie $\prec\!\prec$ na množine úloh $\mathcal{J}$. Ak je nutné pred začiatkom vykonania úlohy J_k ukončiť vykonávanie úlohy J_i, budeme hovoriť, že *úloha J_i predchádza úlohu J_k* a písať $J_i \prec J_k$. *Úloha J_i bezprostredne predchádza úlohu J_k*, ak $J_i \prec J_k$ a neexistuje úloha J_m taká, že $J_i \prec J_m$ a $J_m \prec J_k$. Potom budeme písať $J_i \prec\!\prec J_k$. Digraf $G_{\prec} = (\mathcal{J}, H)$, kde $H = \{(J_i, J_k) \mid J_i \prec J_k\}$, nazveme *precedenčným digrafom precedencie $\prec$ na množine $\mathcal{J}$*. Možno tiež definovať *graf bezprostrednej precedencie na množine $\mathcal{J}$* ako $G_{\prec\!\prec} = (\mathcal{J}, H)$, kde $H = \{(J_i, J_k) \mid J_i \prec\!\prec J_k\}$.

2.1.2 Rozvrhovací problém

Pri rozvrhovacom probléme máme danú množinu úloh $\mathcal{J}$ pozostávajúcu z úloh $J_1, J_2, \ldots J_n$, teda $\mathcal{J} = \{J_1, J_2, \ldots J_n\}$. Každá úloha J_i pozostáva z vykonania niekoľkých operácií $o_{i1}, o_{i2} \ldots, o_{im_i}$, t. j. $J_i = \{o_{i1}, o_{i2} \ldots, o_{im_i}\}$.
Ďalej máme danú množinu strojov $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$. Pre každú operáciu o_{ij} ľubovoľnej úlohy i je daná jej časová dĺžka p_{ij} jej spracovania (processing time) a prípadne aj stroj $\mu_{ij} \in \mathcal{M}$, na ktorom sa môže táto operácia vykonať (ak ide o špecializované stroje).
Rozvrhovací problém považujeme za daný, ak sú dané:

1. Množina strojov $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$ a špecifikácia, či ide o stroje univerzálne, kedy je každý stroj schopný spracovať všetky operácie, alebo o stroje špecializované, kedy je jeden stroj schopný spracovať len operácie jedného typu.

2. Množina úloh $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ a pre každú úlohu J_i príslušná postupnosť operácií $o_{i1}, o_{i2} \ldots, o_{im_i}$. V prípade špecializovaných strojov musí byť naviac dané postupnosť strojov $\mu_{i1}, \mu_{i2} \ldots, \mu_{im_i}$ určujúca, že operácia o_{ij} sa spracuje na stroji μ_{ij}. Posledne menovanej postupnosti sa niekedy hovorí poradie prechodu úlohy J_i strojmi.

3. Precedenčná relácia $\prec$ na množine operácií[1].

[1] Relácia $\prec$ môže byť aj prázdna, t. j. na poradie operácií nie sú kladené žiadne požiadavky.

4. Kriteriálna funkcia rozvrhu.

2.1.3 Rozvrh

Definícia 2.4. *Zostaviť rozvrh pre množinu úloh* $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$, kde úloha J_i pozostáva z operácií $o_{i1}, o_{i2} \ldots, o_{im_i}$, znamená pre každú operáciu o_{ij} určiť časový okamih b_{ij} – *začiatok vykonávania operácie* o_{ij} *(beginning time)* a časový okamih c_{ij} – *koniec vykonávania operácie* o_{ij} *(completion time)*, tak, že

1. $(c_{ij} - b_{ij}) = p_{ij}$ – v rozvrhu vyhradený čas na spracovanie operácie o_{ij} je rovný času p_{ij}, ktorý operácia o_{ij} vyžaduje.

2. Ak o_{ij}, o_{kl} sú dve operácie, pre ktoré je $o_{ij} \prec o_{kl}$, potom $c_{ij} \leq b_{kl}$ – ak operácia o_{ij} predchádza operáciu o_{kl}, potom sa vykonávanie operácie o_{kl} nezačne skôr, než sa ukončí vykonávanie operácie o_{ij}.

2.1.4 Vstupné dáta rozvrhovacieho problému

Pre každú úlohu J_i môžu byť špecifikované tieto dáta:

- počet m_i a množina operácií $\{o_{i1}, o_{i2}, \ldots, o_{im_i}\}$ úlohy J_i.

- v prípade, že ide o špecializované stroje, m_i strojov $\{\mu_{i1}, \mu_{i2}, \ldots, \mu_{im_i}\}$ takých, že operácia o_{ij} sa má vykonať na stroji μ_{ij}.

- m_i časov na spracovanie $p_{i1}, p_{i2}, \ldots, p_{im_i}$, ktoré po rade potrebujú operácie $o_{i1}, o_{i2}, \ldots, o_{im_i}$ na spracovanie na príslušných strojoch, na ktorých majú byť vykonané.
  Ak úloha J_i pozostáva z jedinej operácie, stačí jediný čas p_i.

- r_i najskôr možný začiatok spracovania úlohy J_i. Tento údaj možno chápať ako čas vstupu úlohy J_i do systému (release date).

- d_i požadovaný čas ukončenia úlohy J_i (due date).

- $a_i = d_i - r_i$ maximálna prípustná doba pobytu úlohy J_i v systéme.

- w_i váha úlohy J_i. Vyjadruje jej relatívnu dôležitosť.

2.1.5 Výstupné dáta rozvrhovacieho problému

Výsledkom riešenia rozvrhovacieho problému je rozvrh – množina časových intervalov (b_{ij}, c_{ij}), v ktorých sa majú operácie o_{ij} vykonať. Z týchto údajov možno pre každú úlohu J_i vypočítať nasledujúce veličiny:

- C_i čas ukončenia úlohy J_i (completion time)
- F_i doba pobytu úlohy J_i v systéme
- W_i celková doba čakania úlohy J_i v systéme
- $L_i = C_i - d_i$ časová odchýlka od plánovaného času ukončenia úlohy J_i
- $T_i = \max\{0, L_i = C_i - d_i\}$ oneskorenie úlohy J_i
- $E_i = \max\{0, -L_i\}$ predstih úlohy J_i
- $U_i = 0$, ak $C_i \leq d_i$, inak $U_i = 1$ penalizačná jednotka úlohy J_i

Vidíme, že veličiny $L_i,\ T_i,\ E_i,\ U_i$ sú definované ako funkcie času ukončenia C_i. Dobu pobytu F_i úlohy J_i v systéme možno vyjadriť ako $F_i = C_i - r_i$, kde r_i je čas vstupu úlohy J_i do systému. Ak p_i je čistá doba spracovania úlohy J_i, potom pre jej celkovú dobu čakania v systéme platí $W_i = F_i - p_i = C_i - r_i - p_i$. Všetky vyššie spomenuté výstupné dáta rozvrhovacieho problému sú teda funkciami základnej veličiny C_i.

2.1.6 Kritériá optimality rozvrhovacieho problému

V literatúre sa pre hodnotenie rozvrhov používajú relatívne jednoduché minimalizačné kritériá optimality. Najlepšie bude teda to riešenie, ktoré bude mať minimálnu hodnotu príslušného kritéria.

Jednou skupinou kritérií sú maximá z časov ukončenia jednotlivých úloh (resp. ich časových diferencií, oneskorení, doby pobytu v systéme atď.) Najčastejšie používané kritériá tohto typu sú

$$C_{\max} = \max_i\{C_i\} \qquad L_{\max} = \max_i\{L_i\} \tag{2.1}$$

$$T_{\max} = \max_i\{T_i\} \qquad F_{\max} = \max_i\{F_i\} \tag{2.2}$$

a ich analógie s použitím váh úloh

$$(Cw)_{\max} = \max_i\{C_i w_i\} \qquad (Lw)_{\max} = \max_i\{L_i w_i\} \tag{2.3}$$

$$(Tw)_{\max} = \max_i\{T_i w_i\} \qquad (Fw)_{\max} = \max_i\{F_i w_i\}\ . \tag{2.4}$$

Zvláštnu pozornosť si zasluhuje veličina $C_{\max} = \max_i\{C_i\}$ – táto predstavuje čas ukončenia poslednej úlohy, a teda dobu spracovania celej vstupnej množiny úloh, preto ju budeme volať aj *dĺžka rozvrhu*. V anglickej literatúre sa pre $C_{\max}$ používa termín *makespan*.
Kritérium $C_{\max}$ použijeme v prípade, keď odberateľ zadá istú množinu úloh $\mathcal{J}$, vykonané úlohy odoberie (a zaplatí) až po ukončení poslednej z nich. Analogicky možno nájsť praktickú motiváciu i pre ostatné kritériá z tejto skupiny.

Druhou skupinou kritérií tvoria súčty časov ukončenia jednotlivých úloh (resp. ich časových diferencií, oneskorení, pobytu v systéme atď.)

$$\sum C_i = \sum_{i=1}^{n} C_i \qquad \sum L_i = \sum_{i=1}^{n} L_i \tag{2.5}$$

$$\sum T_i = \sum_{i=1}^{n} T_i \qquad \sum U_i = \sum_{i=1}^{n} U_i \tag{2.6}$$

V prípade nerovnakého významu jednotlivých úloh môžeme použiť ako kritériá vážené sumy predchádzajúcich veličín:

$$\sum w_i C_i = \sum_{i=1}^{n} w_i C_i \qquad \sum w_i L_i = \sum_{i=1}^{n} w_i L_i \tag{2.7}$$

$$\sum w_i T_i = \sum_{i=1}^{n} w_i T_i \qquad \sum w_i U_i = \sum_{i=1}^{n} w_i U_i \tag{2.8}$$

Praktickou motiváciou na použitie týchto kritérií je snaha znižovať náklady spojené s pobytom úloh v systéme (prvé dve kritériá), pokút za oneskorené dodávky, či počet omeškaných úloh.

V niektorej literatúre sa ako kritérium objavuje priemerná hodnota časov ukončenia úloh $J_1, J_2, \ldots, J_n$ označovaná ako $\overline{C}$, či priemerné hodnoty ich časovej diferencie, omeškania, doby pobytu v systéme, čakania atď. označované ako $\overline{L}$, $\overline{T}$, $\overline{F}$, $\overline{W}$ počítané ako

$$\overline{C} = \frac{1}{n}\sum_{i=1}^{n} C_i \quad \overline{L} = \frac{1}{n}\sum_{i=1}^{n} L_i \quad \overline{T} = \frac{1}{n}\sum_{i=1}^{n} T_i \quad \overline{W} = \frac{1}{n}\sum_{i=1}^{n} W_i$$

Keďže počet úloh n je konštanta, je kritérium $\sum C_i$ n–násobkom kritéria $\overline{C}$, a preto sú obe kritériá ekvivalentné. Analogicky sú ekvivalentné $\sum L_i$ s $\overline{L}$, $\sum T_i$ s $\overline{T}$ atď.

Poznamenajme ešte, že

$$\sum_{i=1}^{n} w_i L_i = \sum_{i=1}^{n} w_i (C_i - d_i) = \sum_{i=1}^{n} w_i C_i - \sum w_i d_i,$$

čiže $\sum w_i C_i$ a $\sum w_i L_i$ sa líšia konštantou, a teda sú ekvivalentné.

Ďalej vidíme, že všetky doteraz spomenuté funkcie používané ako kritériá optimality boli v konečnom dôsledku funkciami časových okamihov ukončenia $C_1, C_2, \ldots, C_n$. Teória rozvrhov pracuje i so zložitejšími kritériami tvaru $Z = f(C_1, C_2, \ldots, C_n)$. Medzi nimi zaujímajú zvláštne postavenie tzv. *regulárne kritériá*.

Definícia 2.5. Kritérium $Z = f(C_1, C_2, \ldots, C_n)$ nazveme *regulárnym*, keď je minimalizačné a keď je neklesajúcou funkciou každej svojej premennej $C_1, C_2, \ldots, C_n$.

Ak je teda kritérium f regulárne a platí $Z = f(C_1, C_2, \ldots, C_n)$, $Z' = f(C'_1, C'_2, \ldots, C'_n)$ a $Z < Z'$, potom musí existovať aspoň jedno i, $1 \leq i \leq n$, pre ktoré je $C_i < C'_i$. Naopak, ak $C'_1 \leq C_1$, $C'_2 \leq C_2, \ldots, C'_n \leq C_n$, potom $Z' = f(C'_1, C'_2, \ldots, C'_n) \leq Z = f(C_1, C_2, \ldots, C_n)$.

Ľahko možno ukázať, že $C_{\max}$, $\overline{C}$, $\overline{W}$, $L_{\max}$, $\overline{L}$, $T_{\max}$, $\overline{T}$ patria k triede regulárnych kritérií.

Praktické úlohy však často vyžadujú i zložitejšie kritériá optimality súvisiace s minimalizáciou celkovej ceny spracovania, ktorú možno počítať ako súčet cien spracovania jednotlivých operácií na jednotlivých strojoch plus súčet cien prestavovania strojov medzi jednotlivými operáciami a do ktorej môžu byť zahrnuté i pokuty za neskoré spracovanie dodávky súvisiace s vymenovanými kritériami. Takéto praktické úlohy sú veľmi špecifické, je ťažké pre ne formulovať všeobecný model, a preto sa musia príslušné matematické modely a algoritmy riešenia vypracovávať priamo na mieru daného problému.

2.2 Klasifikácia rozvrhovacích systémov

Čitateľ oboznámený so základmi teórie hromadnej obsluhy pozná trojmiestnu Kendallovu klasifikáciu systémov hromadnej obsluhy, v ktorej napr. symbolom $M|M|1$ označujeme systém s poissonovským prúdom požiadaviek, exponenciálne rozdelenou dobou obsluhy a jednou linkou obsluhy. Podobné klasifikácie existujú aj pre rozvrhovacie problémy. Prvým pokusom bola *Conwayova klasifikácia*, ktorá sa však príliš neujala. Všetky súčasné klasifikačné systémy rozvrhovacích problémov vychádzajú z klasifikácie, ktorú navrhli Lawler, Lenstra a Rinoy Kan a budeme ju označovať ako *klasifikácia LLRK*.

Táto klasifikácia má tri položky $\alpha|\beta|\gamma$.

Položka α charakterizuje množinu strojov $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$. Druhá položka β charakterizuje množinu úloh $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$. Tretia položka γ vyjadruje optimalizačné kritérium.

2.2.1 Prostredie strojov α

Prostredie strojov popisuje prvá položka α notácie $\alpha|\beta|\gamma$. Táto môže mať tvar

$$Pm,\ Qm,\ Rm,\ Om,\ Fm,\ Jm,$$

kde m je parameter označujúci počet strojov. Na jeho mieste môže byť konkrétne číslo (napríklad $1, 2, 3, \ldots$), "∞" vyjadrujúce dostatočne veľký počet strojov, alebo samotný symbol "m" vyjadrujúci ľubovoľný všeobecný počet strojov. V posledne menovanom prípade sa niekedy symbol "m" vynecháva.

Univerzálne stroje

Prvé tri možné hodnoty Pm, Qm, Rm prvého parametra α trojpoložkovej klasifikácie $\alpha|\beta|\gamma$ označujú, že ide o systém s m-prvkovou množinu $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$ strojov, z ktorých každý môže vykonávať každú operáciu. Každá úloha J_i pozostáva v týchto prípadoch z jedinej operácie, (t. j. $J_i = \{o_{i1}\} = \{o_i\}$), ktorá môže byť vykonaná na ľubovoľnom stroji M_j, pričom čas spracovania úlohy J_i (čo je to isté ako čas spracovania operácie o_i) na stroji M_j označíme symbolom p_{ij}. Hovoríme, že ide o prípad paralelných univerzálnych strojov.

Pm – máme m identických paralelných strojov $p_{ij} = p_{i1} = p_i$. Vtedy čas spracovania každej operácie nezávisí od toho, na ktorom stroji je vykonaná.

Qm – máme m uniformných paralelných strojov. Uniformita tu znamená, že všetky stroje sú rovnaké až na rýchlosť. Preto existujú čísla p_i, q_j, $i = 1, \ldots, n$, $j = 1, \ldots, m$ také, že $p_{ij} = p_i.q_j$. Číslo q_j sa volá faktor rýchlosti stroja M_j. V niektorej literatúre sa používa veličina $v_j = \frac{1}{q_j}$ a vyjadruje rýchlosť stroja M_j. Potom $p_{ij} = \frac{p_i}{v_j}$.

Rm – máme m ľubovoľných paralelných strojov a hodnoty p_{ij} sú ľubovoľné.

Ak počet strojov $m = 1$, systémy $P1$, $Q1$, $R1$ sú ekvivalentné, a vtedy namiesto $P1$, resp. $Q1$ či $R1$ píšeme iba symbol 1.

Príklad

Vo výpočtovom laboratóriu je m identických počítačov rovnakého typu – majú rovnaký procesor, rovnakú taktovaciu frekvenciu, rovnakú pamäť, rovnaké periférie. Pre tieto počítače treba rozvrhnúť spracovanie n programov. V tomto prípade pôjde o identické paralelné stroje $Pm|\beta|\gamma$.
Ak by sa počítače líšili len taktovacou frekvenciou (a teda rýchlosťou), išlo by o uniformné paralelné stroje $Qm|\beta|\gamma$.
Ak by naviac počítače neboli rovnaké (napríklad niektoré by mali koprocesor na spracovanie číselných operácií v pohyblivej čiarke a iné nie), išlo by o ľubovoľné paralelné stroje $Rm|\beta|\gamma$.

Špecializované stroje

Ďalšie tri hodnoty parametra α – Om, Fm, Jm sú určené pre situácie, kedy m-prvková množina strojov $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$ obsahuje špecializované stroje, z ktorých každý je schopný spracovať iba jeden typ operácií. Každá úloha $J_i = \{o_{i1}, o_{i2}, \ldots, o_{im}\}$ pozostáva z práve m operácií a každá z týchto operácií vyžaduje práve jeden stroj z množiny $\mathcal{M}$. Symbolom p_{ij} označíme čas spracovania operácie o_{ij} na príslušnom stroji.

Nasledujúce tri hodnoty parametra α znamenajú

Om - máme *Open Shop* s m strojmi, kde každá úloha je tvaru $J_i = \{o_{i1}, o_{i2} \ldots, o_{im}\}$, a kde každá operácia o_{ij} má byť vykonaná na stroji $\mu_j \in \mathcal{M}$ za p_{ij} časových jednotiek, pričom na poradí spracovania operácií v rámci úlohy J_i nezáleží. Vhodnou indexáciou operácií v rámci každej z úloh J_i možno dosiahnuť, že všetky operácie o_{i1} sa spracujú na

stroji M_1, operácie o_{i2} na stroji M_2, atď., všetky operácie o_{im} na stroji M_m,

Fm - máme *Flow Shop* s m strojmi, v ktorom každá úloha je tvaru $J_i = \{o_{i1}, o_{i2} \ldots, o_{im}\}$, pričom $o_{i1} \prec o_{i2} \prec \cdots \prec o_{im}$, a kde operácia o_{i1} sa spracuje na stroji M_1, operácia o_{i2} na stroji M_2, atď. až operácia o_{im} na stroji M_m, pričom trvanie spracovania operácie o_{ij} na stroji M_j je p_{ij}. Inými slovami, pri flow shope má každá úloha určené poradie spracovania operácií, pričom všetky úlohy prechádzajú strojmi v rovnakom poradí.

Jm - máme *Job Shop* s m strojmi, v ktorom každá úloha je tvaru $J_i = \{o_{i1}, o_{i2} \ldots, o_{im}\}$, operácie majú byť vykonané v poradí $o_{i1} \prec o_{i2} \prec \cdots \prec o_{im}$ a každá operácia o_{ij} má byť vykonaná na pridelenom stroji μ_{ij} za p_{ij} časových jednotiek.

Ak $m = 1$, systémy $O1,\ F1,\ J1$ sú ekvivalentné a vtedy namiesto $O1$, resp. $F1$ či $J1$ píšeme iba symbol 1.

Príklady

Výroba ložiskových krúžkov. S každým ložiskovým krúžkom sa musia vykonať nasledujúce operácie: kovanie, sústruženie, kalenie a brúsenie. Tieto operácie sa robia na strojoch zápustkový kovací stroj, sústruh, kaliareň a brúska. Množina operácií s jedným krúžkom predstavuje úlohu. Operácie každej úlohy sú lineárne usporiadané a všetky úlohy potrebujú stroje v rovnakom poradí. Jednotlivé úlohy sa môžu líšiť len časmi spracovania operácií. Preto tu ide o rozvrhovaciu úlohu typu flow shop $Fm|\beta|\gamma$.

Technická kontrola vozidiel. Na technickú kontrolu je objednaných n vozidiel. Každému vozidlu treba skontrolovať emisie, brzdy, svetlá a geometriu prednej nápravy. Práve vymenovaná množina kontrol s jedným vozidlom predstavuje úlohu, každá z vymenovaných kontrol je operáciou. Keďže na poradí kontrol – operácií nezáleží, dostali sme rozvrhovaciu úlohu typu open shop $Om|\beta|\gamma$.

Výroba rôznych výrobkov. V pláne dielne je výroba n kusov rôznych výrobkov. V dielni je vŕtačka, sústruh, brúska a fréza. Všetky výrobky majú stanovené poradie operácií, avšak niektoré prechádzajú strojmi v poradí vŕtačka, sústruh, brúska, fréza, iné v poradí sústruh, fréza, vŕtačka, brúska a iné ešte v inom poradí. Tu ide o rozvrhovaciu úlohu typu job shop $Jm|\beta|\gamma$.

2.2.2 Prostredie úloh β

Druhá položka β popisuje prostredie úloh. Je v tvare postupnosti symbolov $\beta = \beta_1, \beta_2, \ldots$, kde symboly na miestach β_i môžu mať nasledujúci význam:

Ak $\beta_i =$ pmtn, je dovolené *prerušenie (preemption) operácií*, t. j. spracovanie ľubovoľnej operácie môže byť prerušené a obnovené neskôr.
Ak sa v parametri β nevyskytuje žiaden symbol $\beta_i =$ pmtn, prerušenie operácií nie je dovolené.

Ak $\beta_i = r_i$, sú dané okamihy $r_1, r_2, \ldots, r_n$ vstupu úloh $J_1, J_2, \ldots, J_n$ do systému. Inak sa predpokladá, že všetky úlohu vstupujú do systému v čase 0.

Ak $\beta_i =$ prec, na množine úloh $\mathcal{J}$ je špecifikovaná relácia precedencie $\prec$. Ak pre $J_i,\ J_j \in \mathcal{J}$ platí $J_i \prec J_j$, prvá operácia úlohy J_j sa môže začať vykonávať až po skončení spracovania poslednej operácie úlohy J_i.

Ak $\beta_i = s_{ij}$, sú dané prestavovacie časy. Ak sa na tom istom stroji spracováva úloha J_j bezprostredne po úlohe J_i, musí byť do rozvrhu po spracovaní úlohy J_i zaradená prestávka v trvaní s_{ij} na prestavenie stroja.

Ak $\beta_i =$ res, pre každú operáciu sú zadanej požiadavky na zdroje a tiež obmedzenia – kapacity týchto zdrojov (zdroje = resources) – (napr. maximálny dovolený odber elektrického prúdu, vody, plynu, obmedzenia na ľudské zdroje atď.). V tomto prípade nesmie súčet požiadaviek všetkých operácií vykonávaných v tom istom okamihu na tieto zdroje prekročiť dané obmedzenia.

Uvedené možnosti sa môžu ľubovoľne kombinovať – napríklad $\beta = pmtn, r_i, prec$ znamená, že ide o systém, kde je dovolené prerušenie vykonávania operácií, úlohy prídu do systému v nerovnakých časoch a je medzi nimi definovaná precedenčná relácia.
Vymenované možnosti zďaleka nie sú všetky, ktoré sa v literatúre vyskytujú. Praktické požiadavky neustále prinášajú nové situácie, a preto počet a variabilita parametrov v poli prostredia úloh rastie.

Príklady

$1||C_{\max}$ – systém s jedným strojom, v ktorom minimalizujeme čas výstupu poslednej úlohy zo systému.

$P3|pmtn|\sum F_i$ – systém s troma paralelnými identickými strojmi, kde je povolené prerušenie operácií, a v ktorom minimalizujeme celkovú dobu pobytu úloh v systéme.
$1|r_i, prec|L_{\max}$ – systém s jedným strojom s nerovnakými časmi príchodu úloh do systému a s precedenčnou reláciou na množine úloh, v ktorom minimalizujeme maximum z odchýlok od požadovaných časov ukončenia úloh.
$P\infty|prec|C_{\max}$ – systém s neobmedzeným počtom strojov, medzi úlohami je precedenčná relácia, minimalizujeme čas výstupu poslednej úlohy zo systému. Toto je vlastne úloha sieťového plánovania, ktorá sa dá riešiť metódou CPM.
$F2||C_{\max}$ – Flow Shop s dvoma strojmi s minimalizáciou času výstupu poslednej úlohy zo systému. Každá z úloh pozostáva z dvoch operácií, všetky úlohy prechádzajú najprv strojom M_1, potom strojom M_2. Úloha známa ako Johnsonov problém. Pre jej riešenie existuje polynomiálny Johnsonov algoritmus. Analogická úloha s troma strojmi je už NP–ťažká.
$Jm|prec|\sum F_i$ – Job Shop s m strojmi, medzi úlohami existuje precedenčná relácia, minimalizuje sa celková doba pobytu úloh v systéme. Úloha je NP–ťažká.

2.3 Niektoré rozvrhovacie systémy

2.3.1 Systémy s jedným strojom

Predpokladajme, že máme rozvrhovací systém, v ktorom každá úloha $J_i \in \mathcal{J}$ pozostáva z jedinej operácie $o_{i1} = o_i$ a pre každú operáciu je určený jediný stroj, na ktorom sa táto operácia môže vykonávať. Ďalej predpokladajme, že medzi úlohami z $\mathcal{J}$ neexistuje žiadna precedenčná relácia a pokiaľ dve úlohy J_i, J_j nevyžadujú ten istý stroj, spracovanie jednej nijako neovplyvňuje spracovanie druhej. Vtedy sa množina úloh $\mathcal{J}$ rozpadne na disjunktné podmnožiny $\mathcal{J}_1, \ldots, \mathcal{J}_m$ také, že všetky úlohy ležiace v podmnožine $\mathcal{J}_j$ vyžadujú na svoje spracovanie (len) stroj M_j. V takomto prípade je spracovanie úloh z rôznych podmnožín na rôznych strojoch nezávislé a daný rozvrhovací problém sa rozpadá na m nezávislých rozvrhovacích problémov – každý s jedným strojom.

Budeme teda predpokladať, že máme danú n–prvkovú množinu úloh $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$, z ktorých každá pozostáva z jedinej operácie o_i – t. j. $J_i = \{o_i\}$, a že počet strojov v systéme je $m = 1$. Ďalej predpokladáme, že všetky úlohy z množiny $\mathcal{J}$ vstúpia do prázdneho systému naraz v čase 0, t. j. $r_i = 0$ pre všetky $1 \leq i \leq n$. Ďalej predpokladáme, že pre každú úlohu J_i je dané $p_i = p_{i1}$ trvanie jej (jedinej) operácie a d_i požadovaný čas ukončenia.

Pre rozvrhovacie systémy s jedným strojom bez prerušenia operácií je rozvrh **S** v zmysle odseku 2.1.3 (str. 22) určený postupnosťou disjunktných časových intervalov $(b_1, c_1), (b_2, c_2), \ldots, (b_n, c_n)$, kde (b_i, c_i) je interval, v ktorom sa spracováva úloha J_i, pričom dĺžka tohto intervalu $c_i - b_i = p_i$. Pre okamih ukončenia C_i úlohy J_i platí $C_i = c_i$.

Pre rozvrhovacie systémy s jedným strojom s prerušením operácií je rozvrh **S** určený postupnosťou disjunktných časových intervalov (b_{ik}, c_{ik}), $i = 1, 2, \ldots, n$, $k = 1, 2, \ldots, k_i$, kde k_i je počet čiastkových intervalov, v ktorých sa vykonáva úloha J_i. Predpokladáme $b_{ik} < c_{ik} < b_{i(k+1)}$ pre každé $i,\ k$ také, že $1 \leq i \leq n$ a $1 \leq k < k_i$. Musí byť $\sum_{k=1}^{k_i}(c_{ik} - b_{ik}) = p_i$.

Označme W_i celkovú dobu čakania úlohy J_i v systéme, C_i okamih jej ukončenia a F_i dobu jej pobytu v systéme. ($W_i,\ C_i,\ F_i$ sú výstupné hodnoty riešenia rozvrhovacieho problému). Potom $C_i = F_i = p_i + W_i$, pretože $r_i = 0$.

Napriek tomu, že rozvrhovacie problémy s jedným strojom sú relatívne jednoduché, majú svoj význam už tým, že sa tu dajú skúmať rôzne druhy rozvrhov a rôzne druhy rozvrhovacích kritérií a ich výsledky sú dostatočne názorné. Niekoľko výsledkov o týchto systémoch možno použiť i v zložitejších prípadoch buď ako smerovanie bádania, alebo sú základom pre približné riešenie reálnych úloh. Existuje však množstvo reálnych situácií, ktoré možno dobre modelovať systémami s jedným strojom.

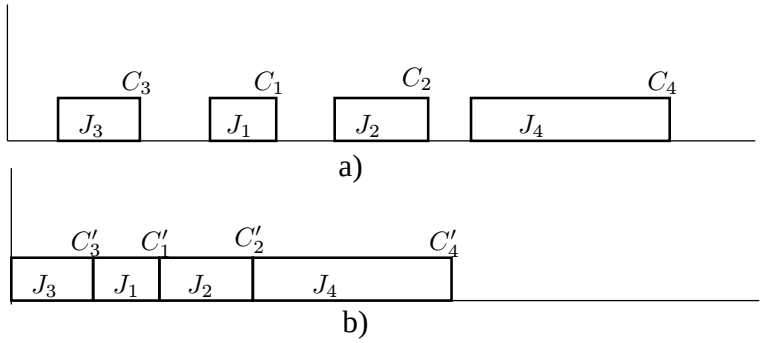
Definícia 2.6. Nech **S** je nejaký rozvrh pre systém s jedným strojom. *Umelý prestoj* v rozvrhu **S** je časový interval (t_1, t_2), v ktorom sa na stroji nevykonáva žiadna úloha, pričom v systéme je úloha čakajúca na spracovanie.

Definícia 2.7. Hovoríme, že $\mathcal{D}$ *je dominantná množina rozvrhov pre kritérium* f, ak v $\mathcal{D}$ existuje aspoň jeden optimálny rozvrh z hľadiska kritéria f.

Veta 2.1. *Majme systém* $1||f$, *resp.* $1|pmtn|f$, *kde* f *je regulárne optimalizačné kritérium. Potom existuje taký optimálny rozvrh* **S** *vzhľadom na kritérium* f, *v ktorom neexistuje umelý prestoj.*

Dôkaz. Princíp dôkazu vidieť z obrázka 2.1. Majme rozvrh **S** s prestojmi a s časmi ukončenia úloh $C_1, C_2, \ldots, C_n$. Ak zrušíme prestoje tým, že posunieme spracovanie úloh skôr, dostaneme nový rozvrh $\mathbf{S}'$, s časmi ukončenia úloh $C'_1, C'_2, \ldots, C'_n$, pre ktoré je $C'_1 \leq C_1, C'_2 \leq C_2, \ldots, C'_n \leq C_n$. Keďže f je regulárne kritérium, musí byť $f(C'_1, C'_2, \ldots, C'_n) \leq f(C_1, C_2, \ldots, C_n)$. ■

Veta 2.2. *Majme systém* $1|pmtn|f$, *kde* f *je regulárne optimalizačné kritérium. Potom existuje taký optimálny rozvrh* **S** *vzhľadom na kritérium* f, *v ktorom niet prerušení operácií.*



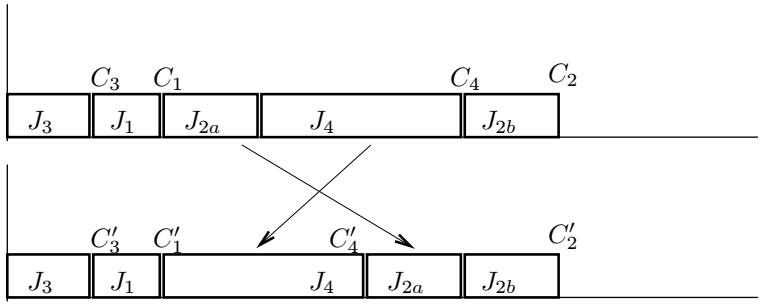
Obr. 2.1: V rozvrhu $\mathbf{S}'$ bez umelých prestojov – diagram b)– skončia všetky úlohy
nie neskôr ako v pôvodnom rozvrhu $\mathbf{S}$ s prestojmi – diagram a)

Dôkaz. Princíp postupného odstraňovania prerušení vidno na obrázku 2.2. Vybratím časti J_{2a} sa vytvorí priestor na posunutie iných úloh skôr. Po tomto posunutí vznikne v rozvrhu miesto na vloženie časti J_{2a} bezprostredne pred časť J_{2b}, čím sa odstráni jedno prerušenie a zároveň sa nezhorší regulárna kriteriálna funkcia nového rozvrhu. ■

Vidíme teda, že optimálne riešenie vzhľadom na regulárne kritérium budeme hľadať medzi rozvrhmi bez prerušenia a bez umelých prestojov. Pre rozvrhovacie problémy bez preempcie pôjde vlastne len o to, v akom poradí naukladáme tesne za sebou intervaly spracovania pre úlohy z danej množiny $\mathcal{J}$. Rozvrh bude daný poradím – permutáciou úloh z $\mathcal{J}$. Preto tieto rozvrhy budeme označovať ako *permutačné rozvrhy*. Počet rôznych permutácií n-prvkovej množiny je $n!$, takže pokusy riešiť rozvrhovacie úlohy tohto typu prezretím všetkých možností sú i pre malé čísla n odsúdené na neúspech. Pre mnohé optimalizačné kritériá však existujú veľmi jednoduché metódy príslušných rozvrhovacích problémov.

Permutácie

Nech $\mathcal{I}$ je množina prirodzených čísel typu $\mathcal{I} = \{1, 2, \ldots, n\}$. Potom *permutáciu* π na množine $\mathcal{I}$ možno pokladať za vzájomne jednoznačné zobrazenie π konečnej množiny $\mathcal{I}$ na seba, čo možno symbolicky zapísať $\pi : \mathcal{I} \to \mathcal{I}$. Prvku $i \in \mathcal{I}$ priradí iný prvok $j \in \mathcal{I}$ podľa predpisu $j = \pi[i]$. Pretože sa permutácia často týka indexov, z dôvodu zjednodušenia zápisu sa často permutácia zapisuje iba ako $j = [i]$ namiesto $j = \pi[i]$. V tomto zmysle budeme používať tento zápis i my.



Obr. 2.2: Úloha J_2 bola vykonávaná v dvoch častiach. Zmenou poradia spracovania častiJ_{2a} a úlohy J_4 dostaneme nový rozvrh $\mathbf{S}'$, v ktorom $C'_4 < C_4$.

Ak máme dané úlohy $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$, zápisom [1] označíme číslo úlohy, ktorá sa vykoná prvá, [2] je číslo úlohy, ktorá sa vykoná na druhom mieste atď. Zápis $[3] = 7$ hovorí, že ako tretia sa vykoná úloha J_7. Podobne $C_{[i]}$ je doba ukončenia úlohy, ktorá sa vykoná ako i–tá. Rozvrh na obrázku 2.1 na strane 32 zodpovedá permutácii $[1] = 3,\ [2] = 1,\ [3] = 2,\ [4] = 4$.

Často sa stane, že budeme potrebovať dve alebo tri permutácie. V tom prípade ich budeme značením rozlišovať čiarkami – $[i]$, $[i]'$ $[i]''$. Pri zložitejších situáciách sa treba vrátiť k pôvodnému označovaniu permutácií $\pi[i]$, $\phi[i]$, $\eta[i], \ldots$, k čomu však v tejto publikácii nedôjde.

Minimalizácia času ukončenia poslednej úlohy – systém $1||C_{\max}$, $1||F_{\max}$

Majme ľubovoľný permutačný rozvrh v systéme $1||f$. Pre časy ukončenia úloh v ľubovoľnom permutačnom rozvrhu platí:

$$C_{[1]} \leq C_{[2]} \leq \cdots \leq C_{[n]},$$

čo nie je nič iné ako samozrejmé tvrdenie, že úloha, ktorá sa spracuje ako prvá, sa ukončí skôr, než úloha spracovaná ako druhá atď. V tomto prípade $b_{[1]} = 0$, $C_{[1]} = c_{[1]} = p_{[1]}$, $C_{[i]} = c_{[i]} = \sum_{k=1}^{i} p_{[k]}$.

Všimnime si, že

$$\begin{aligned} C_{[1]} &= p_{[1]} \\ C_{[2]} &= C_{[1]} + p_{[2]} = p_{[1]} + p_{[2]} \\ &\cdots\cdots \\ C_{[n]} &= C_{[n-1]} + p_{[n]} = \sum_{i=1}^{n} p_{[i]} \end{aligned}$$

$$C_{\max} = C_{[n]} = \sum_{k=1}^{n} p_{[k]} = \sum_{i=1}^{n} p_i$$

Posledná rovnosť platí preto, lebo sčítanie ľubovoľného počtu sčítancov nezávisí od ich poradia. Vidíme, že $C_{\max}$ je pre všetky permutácie rovnaké.

Pretože všetky úlohy prídu do systému naraz v čase 0, sú všetky $r_i = 0$, je

$$F_{[i]} = C_{[i]} - r_{[i]} = C_{[i]} = \sum_{k=1}^{i} p_{[k]}.$$

Vidíme, že $C_{\max} = F_{\max}$ a táto hodnota nezávisí od permutácie $[i]$. Z hľadiska optimalizačného kritéria $C_{\max}$, resp. $F_{\max}$ sú teda všetky permutačné rozvrhy ekvivalentné.

Systémy $1||(Cw)_{\max}$**,** $1||(Fw)_{\max}$
Ak máme pre každú úlohu J_i danú jej váhu w_i vyjadrujúcu jej dôležitosť, môžeme namiesto času ukončenia poslednej úlohy minimalizovať maximum z hodnôt $C_1w_1, C_2w_2, \ldots, C_nw_n$ – toto kritérium sa označuje ako $(Cw)_{\max}$. Pretože všetky úlohy v tomto prípade prídu do systému naraz v čase 0, je toto kritérium totožné s kritériom $(Fw)_{\max}$. Konštrukciu optimálneho permutačného rozvrhu pre skúmaný systém osvetľuje nasledujúca veta.

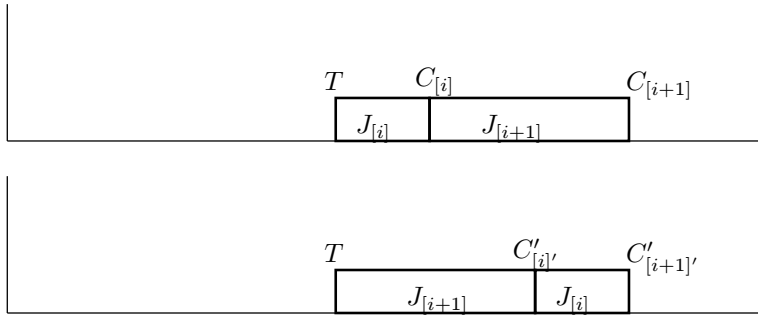
Veta 2.3. *Postačujúcou podmienkou pre optimalitu permutačného rozvrhu* **S** *daného permutáciou* $[\ ]$ *v systéme* $1||(Fw)_{\max}$*, resp.* $1||(Cw)_{\max}$ *je*

$$w_{[1]} \geq w_{[2]} \geq \cdots \geq w_{[n]} \tag{2.9}$$

Dôkaz. Majme optimálny rozvrh **S** určený permutáciou $[\ ]$. Keby existovalo také i, že by $w_{[i]} < w_{[i+1]}$, zostrojme rozvrh $\mathbf{S}'$ s permutáciou $[\ ]'$ definovanou

nasledujúco

$$[j]' = \begin{cases} [j] & \text{ak } j \neq i,\ j \neq i+1 \\ [i+1] & \text{pre } j = i \\ [i] & \text{pre } j = i+1 \end{cases} \tag{2.10}$$



Obr. 2.3: Časť pôvodného rozvrhu **S** a rozvrhu **S′**, ktorý vznikol z rozvrhu **S** zámenou poradia úloh na miestach i, $i+1$. Ostatné časti rozvrhov **S** a **S′** sú identické.

Označme $C_{[j]}$ čas ukončenia úlohy $J_{[j]}$ v rozvrhu **S**, $C'_{[j]'}$ čas ukončenia úlohy $J_{[j]'}$ v rozvrhu **S′** pre $j = 1, 2, \ldots, n$. Platí:

$$C'_{[j]'} = \begin{cases} C_{[j]} & \text{ak } j \neq i \\ T + p_{[i+1]} & \text{pre } j = i \end{cases},$$

kde $T = C_{[i-1]}$, ak $i > 1$ a $T = 0$ pre $i = 1$. Špeciálne $C'_{[i+1]'} = C_{[i+1]}$.
Porovnajme $\max \mathcal{C}$, $\max \mathcal{C}'$, kde $\mathcal{C} = \{C_{[1]}w_{[1]}, C_{[2]}w_{[2]}, \ldots, C_{[n]}w_{[n]}\}$,
$\mathcal{C}' = \{C'_{[1]'}w_{[1]'}, C'_{[2]'}w_{[2]'}, \ldots, C'_{[n]}w_{[n]'}\}$.
Množiny $\mathcal{C}$, $\mathcal{C}'$ sa líšia len v dvoch prvkoch, a to $C_{[i]}w_{[i]}$, $C_{[i+1]}w_{[i+1]}$ a $C'_{[i]'}w_{[i]'}$, $C'_{[i+1]'}w_{[i+1]'}$.
Platí:

$$C'_{[i+1]'}w_{[i+1]'} = C_{[i+1]}w_{[i]} < C_{[i+1]}w_{[i+1]},$$

lebo podľa predpokladu $w_{[i]} < w_{[i+1]}$. Ďalej platí:

$$C'_{[i]'}w_{[i]'} = C'_{[i]'}w_{[i+1]} < C_{[i+1]}w_{[i+1]}.$$

Vidíme, že ku každému prvku množiny $\mathcal{C}'$ existuje väčší alebo rovný prvok množiny $\mathcal{C}$, preto je $\max \mathcal{C}' \leq \max \mathcal{C}$. Keďže podľa predpokladu bol rozvrh $\mathbf{S}$ optimálny, musí byť aj rozvrh $\mathbf{S}'$ určený permutáciou $[\,]'$ optimálny.
Takto postupnými výmenami z optimálneho rozvrhu $\mathbf{S}$ dostaneme optimálny rozvrh $\mathcal{S}'$, kde pre každé $i = 1, 2, \ldots, n-1$ platí $w_{[i]} \geq w_{[i+1]}$.
Nakoniec si ešte všimnime, že ak $w_{[i]} = w_{[i+1]}$, susednými zámenami poradia úloh s rovnakými váhami nepokazíme optimalitu rozvrhu, preto sú všetky permutácie splňujúce (2.9) optimálne. ■

Minimalizácia priemernej doby pobytu úloh

Veta 2.4. *Majme systém* $1||\overline{F}$. *Priemerná doba pobytu úlohy v systéme* $\overline{F}$ *je minimálna práve vtedy, keď*

$$p_{[1]} \leq p_{[2]} \leq \cdots \leq p_{[n]}. \tag{2.11}$$

Dôkaz. Majme rozvrh $\mathbf{S}$ určený permutáciou $[\,]$. Kriteriálnu funkciu $f(\mathbf{S})$ – priemernú dobu $\overline{F}$ pobytu úloh v systéme vypočítame zo vzťahu

$$nf(\mathbf{S}) = n\overline{F} = \sum_{i=1}^{n} F_{[i]} = \sum_{i=1}^{n}\sum_{k=1}^{i} p_{[k]} =$$
$$p_{[1]} + (p_{[1]} + p_{[2]}) + (p_{[1]} + p_{[2]} + p_{[3]}) + \cdots + (p_{[1]} + p_{[2]} + \cdots + p_{[n]}) =$$
$$\sum_{i=1}^{n}(n-i+1)p_{[i]} \tag{2.12}$$

Predpokladajme, že existuje také i, $0 < i < n$, pre ktoré je $p_{[i]} > p_{[i+1]}$. Definujme permutáciu $[\,]'$ predpisom (2.10). Potom

$$nf(\mathbf{S}) - nf(\mathbf{S}') = \sum_{j=1}^{n}(n-j+1)p_{[j]} - \sum_{j=1}^{n}(n-j+1)p_{[j]'} =$$
$$(n-i+1)p_{[i]} + (n-i)p_{[i+1]} - (n-i+1)p_{[i+1]} - (n-i)p_{[i]} = p_{[i]} - p_{[i+1]}. \tag{2.13}$$

Pretože však $p_{[i]} > p_{[i+1]}$, z posledného vzťahu vyplýva, že $p_{[i]} - p_{[i+1]} > 0$, a teda $f(\mathbf{S}') < f(\mathbf{S})$. Za predpokladu, že máme také i, pre ktoré je $p_{[i]} > p_{[i+1]}$ sme zostrojili rozvrh $\mathbf{S}'$ s menšou hodnotou kriteriálnej funkcie.

Na to, že z (2.11) vyplýva optimalita rozvrhu $\mathbf{S}$, stačí ukázať, že ak pre iný rozvrh $\mathbf{S}'$ s permutáciou $[i]'$ platí:

$$p_{[1]'} \leq p_{[2]'} \leq \cdots \leq p_{[n]'},$$

potom $f(\mathbf{S}) = f(\mathbf{S}')$. Permutáciu $\mathbf{S}'$ však možno dostať z permutácie $\mathbf{S}$ aplikovaním konečného počtu susedných zámen na miestach k, $(k+1)$ takých, že $p_{[k]} = p_{[k+1]}$. Zmenu účelovej funkcie takejto zámeny popisuje vzťah (2.13), ktorého pravá strana bude v tomto prípade rovná 0. Keďže sa hodnota kriteriálnej funkcie zachová pri každej susednej výmene, zachová sa i pri výslednej permutácii $[i]'$ definujúcej rozvrh $\mathbf{S}'$. ■

Minimalizácia váženého súčtu dôb pobytu

Veta 2.5. *Majme systém* $1||\overline{Fw}$. *Súčet vážených dôb pobytu v systéme* $\overline{Fw}$ *je minimálny práve vtedy, keď*

$$\frac{p_{[1]}}{w_{[1]}} \leq \frac{p_{[2]}}{w_{[2]}} \leq \cdots \leq \frac{p_{[n]}}{w_{[n]}} \tag{2.14}$$

Dôkaz. Majme rozvrh $\mathbf{S}$ daný permutáciou $[\ ]$, v ktorom neplatí (2.14). Nech existuje $0 < i < n$ také, že

$$\frac{p_{[i]}}{w_{[i]}} > \frac{p_{[i+1]}}{w_{[i+1]}}.$$

Definujme rozvrh $\mathbf{S}'$ s permutáciou $[\ ]'$, pre ktorú je $[i]' = [i+1]$, $[i+1]' = [i]$ a $[j]' = [j]$ pre $j \neq i$, $j \neq i+1$. Nech $f(\mathbf{S}) = \sum_{i=1}^{n} F_{[i]}.w[i]$. Potom pre dobu pobytu úlohy $[i]$ $F_{[i]}$ podľa rozvrhu $\mathbf{S}$ a dobu pobytu úlohy $[i]'$ $F'_{[i]'}$ podľa rozvrhu $\mathbf{S}'$ platí:

$$F'_{[j]'} = F_{[j]} \qquad \text{pre } j \neq i.$$

Pre $j = i$ platí:

$$F'_{[i]'} = F_{[i-1]} + p_{[i]'} = F_{[i-1]} + p_{[i+1]} = F_{[i]} - p_{[i]} + p_{[i+1]} \tag{2.15}$$

$$\begin{aligned} f(\mathbf{S}) - f(\mathbf{S}') &= \sum_{j=1}^{n} F_{[j]}.w_{[j]} - \sum_{j=1}^{n} F_{[j]'}.w_{[j]'} = \\ &= F_{[i]}.w_{[i]} + F_{[i+1]}.w_{[i+1]} - F'_{[i]'}.w_{[i]'} - F'_{[i+1]'}.w_{[i+1]'} = \\ &= F_{[i]}.w_{[i]} + F_{[i+1]}.w_{[i+1]} - F'_{[i]'}.w_{[i+1]} - F'_{[i+1]'}.w_{[i]} = \\ &= w_{[i+1]}(F_{[i+1]} - F'_{[i]'}) - w_{[i]}(F'_{[i+1]'} - F_{[i]}) = w_{[i+1]}.p_{[i]} - w_{[i]}.p_{[i+1]}. \end{aligned} \tag{2.16}$$

Ak $\frac{p_{[i]}}{w_{[i]}} > \frac{p_{[i+1]}}{w_{[i+1]}}$, je $f(\mathbf{S}) - f(\mathbf{S}') > 0$. Ak $\frac{p_{[i]}}{w_{[i]}} = \frac{p_{[i+1]}}{w_{[i+1]}}$, je $f(\mathbf{S}) - f(\mathbf{S}') = 0$.
Z dokázaného vyplýva:

1. Ak nie je splnená podmienka (2.14), príslušný rozvrh $\mathbf{S}$ nie je optimálny z hľadiska kritéria $\overline{Fw}$.

2. Ak pre permutácie rozvrhov $\mathbf{S}$, $\mathbf{S}'$ príslušné permutácie splňujú podmienku (2.14), potom oba rozvrhy majú tú istú hodnotu kriteriálnej funkcie $\overline{Fw}$.

■

Minimalizácia maxima omeškania, resp. maxima časovej diferencie

Veta 2.6. *Majme systém* $1||T_{\max}$*, resp.* $1||L_{\max}$*,* $\mathbf{S}$ *permutačný rozvrh. Na to, aby maximum omeškania* $T_{\max} = \max_{1\leq i\leq n}\{T_i\}$*, resp. maximum časových diferencií* $L_{\max} = \max_{1\leq i\leq n}\{L_i\}$ *jednotlivých úloh bolo minimálne, stačí, aby*

$$d_{[1]} \leq d_{[2]} \leq \cdots \leq d_{[n]} \ .$$

Dôkaz. Časová diferencia úlohy i je definovaná ako $L_i = C_i - d_i$. Pre poslednú úlohu $J_{[n]}$ podľa rozvrhu $\mathbf{S}$ je $L_{[n]} = C_{[n]} - d_{[n]}$. Ako už bolo ukázané, hodnota $C_{[n]} = C_{\max}$ nezávisí od permutácie $[i]$ - je pre všetky rozvrhy rovnaká. Ak chceme minimalizovať hodnotu $C_{[n]} - d_{[n]}$, musíme ako poslednú zaradiť tú úlohu J_i, ktorá má najväčší plánovaný čas ukončenia d_i. Usporiadanie podľa (1.18) tejto podmienke vyhovuje.
Pre kritérium $T_{\max}$ je dôkaz analogický. ■

Minimalizácia váženého súčtu časových diferencií

Skúmajme teraz kritériá $\overline{L} = \sum_{i=1}^{n} L_i$ a $\overline{Lw} = \sum_{i=1}^{n} L_i.w_i$
Platí:

$$\overline{L} = \sum_{i=1}^{n} L_i = \sum_{i=1}^{n}(C_i - d_i) = \sum_{i=1}^{n}(F_i - d_i) = \sum_{i=1}^{n} F_i - \sum_{i=1}^{n} d_i$$

$$\overline{Lw} = \sum_{i=1}^{n} L_i.w_i = \sum_{i=1}^{n}(C_i - d_i).w_i = \sum_{i=1}^{n}(F_i - d_i).w_i = \sum_{i=1}^{n} F_i.w_i - \sum_{i=1}^{n} d_i.w_i$$

Keďže $\sum_{i=1}^{n} d_i$ vo vzťahu (1.19), resp. $\sum_{i=1}^{n} d_i.w_i$ vo vzťahu (1.20) sú konštanty, kritérium $\overline{L}$ je minimálne práve vtedy, keď je minimálne kritérium $\overline{F}$, kritérium $\overline{Lw}$ je minimálne práve vtedy, keď je minimálne kritérium $\overline{Fw}$.
S použitím viet 2.4, 2.5 (str. 36, 37) možno teda tvrdiť:

Veta 2.7. *Majme systém* $1||\overline{L}$. *Priemerná časová diferencia úloh v systéme* $\overline{L}$ *je minimálna práve vtedy, keď*

$$p_{[1]} \leq p_{[2]} \leq \cdots \leq p_{[n]} \ .$$

Veta 2.8. *Majme systém* $1||\overline{Lw}$. *Súčet* $\overline{Lw}$ *vážených časových diferencií úloh v systéme je minimálny práve vtedy, keď*

$$\frac{p_{[1]}}{w_{[1]}} \leq \frac{p_{[2]}}{w_{[2]}} \leq \cdots \leq \frac{p_{[n]}}{w_{[n]}} \ .$$

Minimalizácia váženého maxima časovej odchýlky, resp. omeškania

Pre kritériá $(Lw)_{\max} = \max_i\{L_i.w_i\}$ a $(Tw)_{\max} = \max_i\{T_i.w_i\}$ možno nájsť optimálny rozvrh nasledujúcim algoritmom:

Algoritmus 2.1. Lawlerov algoritmus
Označme $\mathcal{I}$ množinu ešte nezaradených úloh, $p = \sum_{i\in\mathcal{I}} p_i$ celková doba spracovania nezaradených úloh.

Krok 1: Polož $p := \sum_{i=1}^{n} p_i$, $\mathcal{I} := \{1, 2, \ldots, n\}$, $k := n$.

Krok 2: Polož j rovné tomu $i \in \mathcal{I}$, pre ktoré je $w_i(p - d_i)$ (pre kritérium $(Lw)_{\max}$), resp. $\max(0, w_i(p - d_i)$ (pre kritérium $(Tw)_{\max}$) minimálne.

Krok 3. Polož $[k] := j$, $\mathcal{I} := \mathcal{I} - \{j\}$, $k := k - 1$, $p := p - p_j$.

Krok 4. Ak $k = 0$ STOP, inak GOTO Krok 2.

Minimalizácia priemerného omeškania

Pre rozvrhovaciu úlohu $1||\overline{T}$, kde $\overline{T} = \sum_{i=1}^{n} T_i$ ($T_i = \max(0, C_i - d_i)$ dlho nebol známy polynomiálny algoritmus, ani sa nedarilo nič dokázať o jej zložitosti. Tejto úlohe bolo venované enormné úsilie, kým sa v r. 1989 podarilo dokázať, že je NP-ťažká. Vieme však ukázať niektoré jej vlastnosti.

Veta 2.9. *Pre rozvrhovaciu úlohu* $1||\overline{T}$ *platí:*

a) Ak permutačný rozvrh **S** *s vlastnosťou* $d_{[1]} \leq d_{[2]} \leq \cdots \leq d_{[n]}$ *obsahuje nanajvýš jednu omeškanú úlohu, potom je* **S** *optimálny rozvrh.*

b) Ak $d_1 = d_2 = \cdots = d_n$*, potom na to, aby bol rozvrh* **S** *optimálny, stačí, aby pre príslušnú permutáciu platilo:*

$$p_{[1]} \leq p_{[2]} \leq \cdots \leq p_{[n]} \ .$$

c) Ak pre permutačný rozvrh **S** *s vlastnosťou* $p_{[1]} \leq p_{[2]} \leq \cdots \leq p_{[n]}$ *sú všetky úlohy omeškané, potom je* **S** *optimálny rozvrh.*

Rozvrhovacia úloha $1||\overline{Tw}$, kde $\overline{Tw} = \sum_{i=1}^{n} T_i.w_i$, patrí medzi NP–ťažké úlohy.

Minimalizácia počtu omeškaných úloh – problém $1||\sum U_i$

Ako poslednú úlohu permutačných rozvrhov uvedieme rozvrhovaciu úlohu, pri ktorej hľadáme rozvrh, ktorý minimalizuje počet omeškaných úloh.

Algoritmus 2.2. Moorov algoritmus na konštrukciu rozvrhu s minimom omeškaných úloh

Krok 1: Vytvor permutačný rozvrh, pre ktorý je

$$d_{[1]} \leq d_{[2]} \leq \cdots \leq d_{[n]}$$

Vytvor postupnosti $E := \{d_{[1]}, d_{[2]}, \ldots, d_{[n]}\}$, $L := \{\ \}$. Polož aktuálny počet členov postupnosti E $n_E := n$.

Krok 2: Ak žiadna úloha z E nie je omeškaná, postupnosť úloh z E zreťazená s ľubovoľným poradím úloh z L dáva optimálne riešenie. Počet prvkov postupnosti L určuje počet omeškaných úloh. STOP.
Inak pokračuj Krokom 3.

Krok 3: V postupnosti E nájdi prvú omeškanú úlohu $[k]$ (je to úloha na k–tom mieste v E). Medzi prvými k úlohami z E nájdi úlohu $[j]$, pre ktorú je doba spracovania p_j maximálna. Presuň prvok j z postupnosti E do množiny L. Uprav patrične permutáciu $[i]$ ($x := [j]$, pre $i = j+1, j+2, \ldots, n_E$ polož $[i] := [i+1]$, $[n_E] := x$, $n_E := n_E - 1$. Potom $E = \{[1], [2], \ldots, [n_E]\}$, $L = \{[n_E + 1], [n_E + 2], \ldots, [n]\}$. GOTO Krok 2.

Rozvrhy so započítaním prestavovacích časov

V prechádzajúcich rozvrhovacích úlohách sme ticho predpokladali, že stroj môže spracovávať ďalšiu úlohu bezprostredne po predchádzajúcej úlohe bez akéhokoľvek prestavovania. Vo všeobecnosti však zmena úlohy znamená i prestavenie stroja.

Veľmi často je čas na prestavenie stroja medzi úlohami nezávislý od typu ukončenej a začínajúcej úlohy. V takomto prípade sa tento prestavovací čas jednoducho zahrnie do času vykonávania každej úlohy p_i. S takto modifikovanými časmi vykonávania úloh už možno použiť všetky doteraz diskutované rozvrhovacie modely.

Sú však prípady, keď z charakteru vykonávania úloh vyplývajú prestavovacie doby, ktoré sú závislé od dvojice ukončovanej a začínajúcej úlohy.

Conway uvádza ako príklad takéhoto problému miešačku farieb. V stroji na výrobu farieb sa vyrábajú farby Biela, Žltá, Modrá, Červená. Po ukončená výroby jednej farby sa musí stroj očistiť od zvyškov predchádzajúcej farby. Zvyšky žltej farby neovplyvnia tak výrobu červenej, ako zvyšky modrej výrobu bielej. Preto dôkladnosť čistenia stroja – a teda aj nastavovacia doba – závisí od predchádzajúcej a nasledujúcej farby. Nastavovacie doby možno zadať vo forme tabuľky

Matica prestavovacích časov

	B	Ž	M	Č
B	0	1	2	3
Ž	6	0	1	2
M	8	6	0	1
Č	9	8	6	0

Hľadajme najprv optimálne cyklické poradie miešania farieb. Existuje $(4-1)! = 6$ rôznych výrobných cyklov, ktoré majú podľa tabuľky všeobecne rôzne celkové súčty nastavovacích dôb:

1.	B — Ž — Č — M — B	$1+2+6+8=17$
2.	B — Ž — M — Č — B	$1+1+1+9=12$
3.	B — Č — Ž — M — B	$3+8+1+8=20$
4.	B — Č — M — Ž — B	$3+6+6+6=21$
5.	B — M — Ž — Č — B	$2+6+2+9=19$
6.	B — M — Č — Ž — M	$2+1+8+6=17$

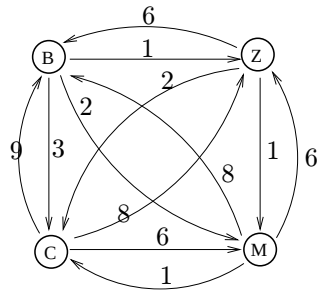
Tu sa už nastavovacie doby nedajú jednoducho zahrnúť do vykonávacích časov jednotlivých úloh. Existencia nastavovacích časov tu už mení charakter rozvrhovacieho procesu. Označme s_{ij} nastavovací čas pri prechode stroja od úlohy J_i k úlohe J_j pre $i, j = 1, 2, \ldots, n$. Naviac označme s_{0i} dobu nastavenia stroja z počiatočného stavu na vykonávanie úlohy J_i.
Pre časy ukončenia spracovania (resp pre doby pobytu) jednotlivých úloh platí:

$$\begin{aligned} C_{[1]} &= F_{[1]} = s_{0[1]} + p_{[1]} \\ C_{[2]} &= F_{[2]} = F_{[1]} + s_{[1][2]} + p_{[2]} \\ &\ldots \\ C_{[n]} &= F_{[n]} = F_{[n-1]} + s_{[n-1][n]} + p_{[n]} \end{aligned}$$

Potom:

$$C_{\max} = F_{\max} = \sum_{i=1}^{n} s_{[i-1][i]} + \sum_{i=1}^{n} p_{[i]} \ .$$

Keďže posledná suma je nezávislá od permutácie i, optimálny permutačný rozvrh z hľadiska kritéria $C_{\max}$, resp. $F_{\max}$ bude ten, ktorý minimalizuje $\sum_{i=1}^{n} s_{[i-1][i]}$.



Obr. 2.4: Model teórie grafov pre systém $1|s_{ij}|C_{\max}$

Situáciu možno modelovať úplným hranovo ohodnoteným digrafom G, v ktorom je každá úloha J_i vrcholom a každá orientovaná hrana (J_i, J_j) je ohodnotená prestavovacím časom s_{ij} – pozri obr. 2.4. Úloha minimalizácie sumárnej doby nastavovania stroja sa teraz dá formulovať ako hľadanie najkratšieho orientovaného cyklu v grafe G obsahujúceho všetky vrcholy a je ekvivalentná so známou úlohou obchodného cestujúceho. Touto úlohou sa podrobnejšie zaoberáme v časti 3.6 na str. 100.

Všeobecný prípad, kedy sa nemusí spracovanie úloh cyklicky opakovať, možno previesť na cyklický prípad zavedením fiktívnej štartovacej a finálnej operácie S s nulovým trvaním a prestavovacími dobami typu s_{Sj} nulovými a dobami typu s_{iS} zodpovedajúcimi úplnému vyčisteniu miešačky tak, aby po ňom mohlo bez zdržania nasledovať miešanie ktorejkoľvek farby. Zodpovedá to podmienke, že začínať i končiť treba s úplne čistou miešačkou.

Nerovnaký príchod úloh do systému

Doteraz sme predpokladali, že všetky úlohy prišli do systému naraz v čase $r_1 = r_2 = \cdots = r_n = 0$. Teraz budeme predpokladať, že $r_i \geq 0$ pre $1 \leq i \leq n$. Ak pre systémy s príchodom všetkých úloh naraz bola dominantnou množinou rozvrhov množina všetkých rozvrhov bez prerušenia a bez umelých prestojov, v tomto prípade to už tak nie je.

Ak je rozvrh **S** bez prerušenia reprezentovaný permutáciou $[i]$, potom pre časy ukončenia $C_{[i]}$ platí:

$$C_{[1]} = r_{[1]} + p_{[1]}$$

$$C_{[2]} = \max\left\{C_{[1]}, r_{[2]}\right\} + p_{[2]}$$

$$C_{[i]} = \max\left\{C_{[i-1]}, r_{[i]}\right\} + p_{[i]}$$

$$C_{[n]} = \max\left\{C_{[n-1]}, r_{[n]}\right\} + p_{[n]}$$

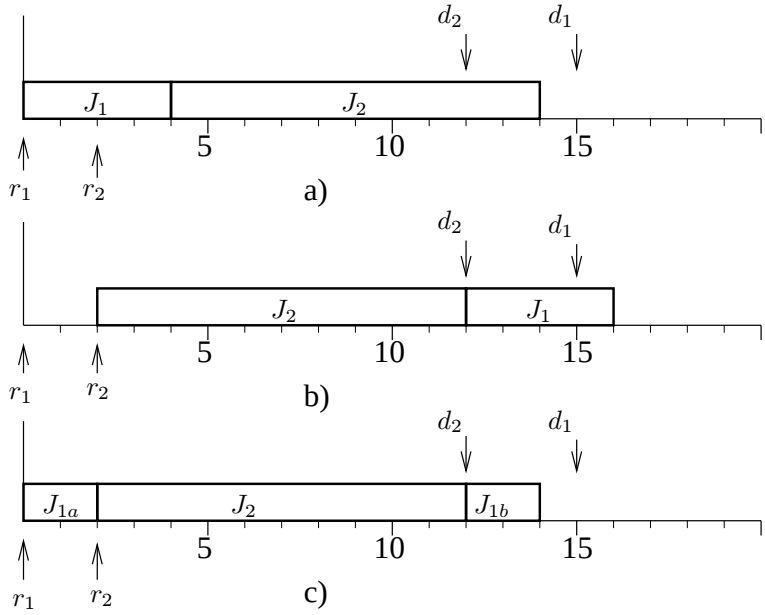
Príklad. Nech $r_1 = 0, p_1 = 4, d_1 = 15$, $r_2 = 2, p_2 = 10, d_2 = 12$. Uvažujme regulárne kritérium $\sum T$, resp. $T_{\max}$.

Pre poradie vykonávania (J_1, J_2) (obrázok 2.5 a)) je $C_1 = 4$, $C_2 = 14$, $T_1 = 0$, $T_2 = 2$, pre poradie (obrázok 2.5 b)) (J_2, J_1) je $C_1 = 16$, $C_2 = 12$, $T_1 = 1$, $T_2 = 0$. Optimálny je teda rozvrh (J_2, J_1), v ktorom je umelý prestoj v čase $\langle 0, r_2)$.

Ak dovolíme prerušenie, potom rozvrh, v ktorom sa najprv vykonajú dve jednotky z úlohy J_1, potom celá úloha J_2 a nakoniec zvyšok úlohy J_1 bude mať $C_1 = 14, C_2 = 12, T_1 = T_2 = 0$ – obr.2.5 c). Najlepší je teda rozvrh s prerušením.

Veta 2.10. *V systéme* $1|r_i|C_{\max}$ *s nerovnakými príchodmi úloh pre optimalitu rozvrhu* **S** *stačí, aby*

$$r_{[1]} \leq r_{[2]} \leq \cdots \leq r_{[n]}.$$



Obr. 2.5: Možné poradia úloh, ak sú dané časy príchodu úloh do systému

Dôkaz. Nech pre niektoré k $1 \leq k < n$ je $r_{[k]} > r_{[k+1]}$. Úloha $J_{[k]}$ sa môže začať spracovávať najskôr v čase $\max\left\{C_{[k-1]}, r_{[k]}\right\}$. V tom je už ale v systéme úloha $J_{[k+1]}$, ktorá by mohla začať už skôr v čase $\max\left\{C_{[k-1]}, r_{[k+1]}\right\}$. Ak by sme len zamenili poradie vykonávania úloh $J_{[k]}, J_{[k+1]}$ na $J_{[k+1]}, J_{[k]}$ s tým, že začiatok spracovania úlohy $J_{[k+1]}$ v novom rozvrhu začne práve vtedy, kedy začalo spracovanie úlohy $J_{[k]}$ v pôvodnom rozvrhu, nezmenia sa hodnoty $C_{[k+2]}, C_{[k+3]}, \ldots, C_{[n]}$. Začiatok spracovania úlohy $J_{[k+1]}$ však možno v novom rozvrhu posunúť do najskôr možného časového okamihu $\max\left\{C_{[k-1]}, r_{[k+1]}\right\}$, čím sa môže vytvoriť možnosť posunúť spracovanie niektorých úloh $J_{[k+1]}, J_{[k+2]}, \ldots, J_{[n]}$ do skorších časových intervalov, a tak prípadne znížiť hodnotu $C_{[n]}$.

Sériou susedných výmen tak dokážeme ku každému rozvrhu $\mathbf{S}$ vytvoriť rozvrh $\mathbf{S}'$ s permutáciou $[\,]'$ $r_{[1]'} \leq r_{[2]'} \leq \cdots \leq r_{[n]'}$, pre ktorý platí $C_{\max}(\mathbf{S}') \leq C_{\max}(\mathbf{S}')$. ■

Ak prerušenie úloh nie je dovolené, je hľadanie optimálneho rozvrhu pre systémy $1|r_i|f$, kde f je jedno z kritérií $L_{\max}$, $T_{\max}$, NP-ťažkým problémom.

Ďalej sa budeme zaoberať systémami $1|r_i, pmtn|f$ s nerovnakým príchodom úloh a s povoleným prerušením úloh.

Algoritmus 2.3. Algoritmus pre hľadanie optimálneho rozvrhu v systémoch $1|r_i, pmtn|L_{\max}$, $1|r_i, pmtn|T_{\max}$

Krok 0: Označ pre každú úlohu J_i τ_i čas potrebný na spracovanie jej zvyšku. Inicializačne polož $\tau_i = p_i$.

Krok 1: Označ $\mathcal{E} \subseteq \mathcal{J}$ množinu všetkých nedokončených úloh s rovnakým najskôr možným začiatkom a_1. Nech a_2 je ďalší najskôr možný začiatok spracovania nejakej ešte neukončenej úlohy. Ak všetky neukončené úlohy môžu začať v čase a_1, $a_2 =: \infty$.

Krok 2: Z množiny $\mathcal{E}$ vyber úlohu J_i, pre ktorú je $d_i = \min_{j\in\mathcal{E}} d_j$. Urči $L = min\{p_i, (a_2 - a_1)\}$ a do intervalu $\langle a_1, L)$ zaraď spracovanie úlohy J_i.

Krok 3: Ak je spracovanie úlohy J_i ukončené, vyraď ju zo zoznamu nedokončených úloh. Ak úloha J_i nie je ešte ukončená, polož $\tau_i =: \tau_i - L$.

Krok 4: Ak sú úplne zaradené všetky úlohy, STOP. Inak GOTO Krok 1.

Predchádzajúci algoritmus možno použiť i pre konštrukciu optimálneho rozvrhu s kritériom $\overline{C}$, resp. $\overline{F}$, ak v Kroku 2 vyberáme za úlohu J_i tú, pre ktorú platí $p_i = \min_{j\in\mathcal{E}} p_j$.

Precedenčná relácia na množine úloh

Medzi úlohami pre jeden stroj môže existovať neprázdna precedenčná relácia $\prec$. Uvedieme zovšeobecnenie Lawlerovho algoritmu pre systém $1|prec|f$ pre systém s rovnakými príchodmi, neprázdnou precedenčnou reláciou $\prec$ a kritériom $(Tw)_{\max}$, $(Lw)_{\max}$, $(Fw)_{\max}$, resp. $(Cw)_{\max}$. Optimálny rozvrh budeme hľadať medzi permutačnými rozvrhmi bez prerušenia a bez umelých prestojov.

Algoritmus 2.4. Zovšeobecnený Lawlerov algoritmus pre systém $1|prec|(Lw)_{\max}$.

Označme $\mathcal{I}$ množinu ešte nezaradených úloh. Ďalej označme $\mathcal{I}_0 \subseteq \mathcal{I}$ podmnožinu takých nezaradených úloh, ktoré nemajú v $\mathcal{I}$ následníkov.

Krok 1: Polož $p := \sum_{i=1}^{n} p_i$, $\mathcal{I} := \{J_1, J_2, \ldots, J_n\}$, nech $\mathcal{I}_0$ je množina úloh, ktoré nemajú následníkov, $k := n$.

Krok 2: Vezmi to $J_i \in \mathcal{I}_0$, pre ktoré je

$$w_i(p - d_i) = \min_{j \in \mathcal{I}_0}\{w_j(p - d_j)\}. \tag{2.17}$$

Krok 3: Polož $[k] := i$, $\mathcal{I} := \mathcal{I} - \{J_i\}$, nech $\mathcal{I}_0$ je množina tých úloh z $\mathcal{I}$, ktoré nemajú v $\mathcal{I}$ následníka, $k := k - 1$, $p := p - p_i$.

Krok 4: Ak $k = 0$, STOP, inak GOTO Krok 2.

Ak v kroku 2 algoritmu 2.4 vyberieme tú úlohu $J_i \in \mathcal{I}_0$, pre ktorú je

$$\max\{0, w_i(p - d_i)\} = \min_{j \in \mathcal{I}_0}\big\{\max\{0, w_j(p - d_j)\}\big\},$$

dostaneme algoritmus pre konštrukciu optimálneho rozvrhu v systéme $1|prec|(Tw)_{\max}$. Ak v druhom kroku algoritmu (2.17) vyberieme tú úlohu $J_i \in \mathcal{I}_0$, pre ktorú je

$$w_i = \min_{j \in \mathcal{I}_0}\{w_j\},$$

algoritmus bude hľadať optimálny rozvrh pre systém $1|prec|(Cw)_{\max}$, resp. $1|prec|(Fw)_{\max}$.

Veta 2.11. *Pre systém* $1|prec|L_{\max}$*, resp.* $1|prec|T_{\max}$ *pre optimalitu rozvrhu* **S** *stačí, aby pre príslušnú permutáciu platilo:*

$$d'_{[1]} \le d'_{[2]} \le \cdots \le d'_{[n]} \ ,$$

kde

$$d'_i = \min\big\{d_i, \min\{d_k \mid i \prec k\}\big\},$$

t. j. d'_i je minimum z požadovaných časov ukončenia úlohy J_i a jej všetkých následníkov.

Problém $1|r_i|L_{\max}$**,** $1|r_i, prec|L_{\max}$

Problém $1|r_i|L_{\max}$ je mimoriadne dôležitý preto, lebo sa často vyskytuje ako podproblém pri heuristických algoritmoch pre job shop a flow shop problém. Napriek tomu, že ide o NP-ťažký problém, existuje pomerne efektívny postup riešenia metódou vetiev a hraníc.

Pri tejto metóde tvoríme permutačný rozvrh postupne od začiatku. Pritom postup modelujeme stromom, ktorého koreň (vrchol úrovne 0) predstavuje prázdny parciálny rozvrh a vetví sa na n vrcholov úrovne 1, ktoré modelujú

jednoúlohové parciálne rozvrhy. Každý z vrcholov úrovne 1 sa vetví na $n-1$ vrcholov úrovne 2, ktorých je $n.(n-1)$ a ktoré predstavujú dvojúlohové parciálne rozvrhy atď. Na úrovni $k-1$ je $n.(n-1).\ldots.(n-k-1)$ vrcholov predstavujúcich $(k-1)$-členné parciálne rozvrhy a z ktorých sa každý vrchol vetví na $n-k$ vrcholov úrovne k.

Ak by sme na k-te miesto parciálneho rozvrhu zaradili doteraz nezaradenú úlohu J_i, táto môže začať najskôr v čase $\max\{C_{[k-1]}, r_i\}$ a skončiť môže najskôr v čase $\max\{C_{[k-1]}, r_i\}+p_i$. Najmenší možný čas ukončenia doteraz nezaradenej úlohy je teda

$$K^* = \min_{i\in\mathcal{I}} \left\{\max\{C_{[k-1]}, r_i\} + p_i\right\}, \tag{2.18}$$

kde $\mathcal{I}$ predstavuje množinu ešte nezaradených úloh a kde $C_{[k-1]}$ je čas ukončenia úlohy $J_{[k-1]}$. Nech J^* je úloha, ktorá minimalizuje (2.18), t. j.

$$\max\{C_{[k-1]}, r^*\} + p^* = \min_{i\in\mathcal{I}} \left\{\max\{C_{[k-1]}, r_i\} + p_i\right\}, \tag{2.19}$$

Ak vrchol na úrovni $k-1$ predstavuje parciálny rozvrh $J_{[1]}, J_{[2]}, \ldots, J_{[k-1]}$, treba uvažovať rozvetvenie do $J_{[1]}, J_{[2]}, \ldots, J_{[k-1]}, J_q$ len vtedy, keď

$$r_q < K^* = \min_{i\in\mathcal{I}} \left\{\max\{C_{[k-1]}, r_i\} + p_i\right\}. \tag{2.20}$$

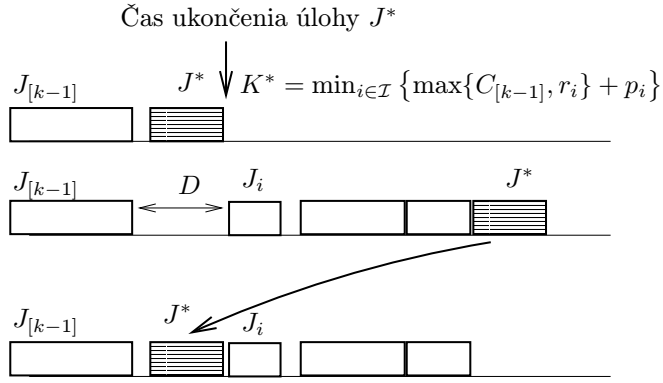
Ak by sme na k-te miesto rozvrhu zaradili úlohu, ktorá nespĺňa (2.20), dostali by sme rozvrh, v ktorom sa pred úlohu na k-tom mieste dá vsunúť ešte úloha J^* bez zmeny časovej polohy ostatných úloh, čím dostaneme rozvrh, ktorý nie je horší než **S**. Táto vlastnosť vo väčšine prípadov značne zjednodušuje vetvenie.

Pre každý vrchol stromu – parciálny rozvrh $J_{[1]}, J_{[2]}, \ldots, J_{[k]}$ je treba ešte určiť dolnú hranicu kritéria $L_{\max}$, ktorú určíme ako maximum hodnoty $L_{\max}$ pre parciálny rozvrh a hodnoty $L_{\max}$ pre ostávajúce nezaradené úlohy pri povolení prerušenia vykonávania operácií – čo vedie k riešeniu systému $1|r_i, pmtn|L_{\max}$, pre ktorý je v tejto kapitole uvedený exaktný algoritmus riešenia.

Na prvý pohľad by sa mohlo zdať, že problém $1|r_i, prec|L_{\max}$ je ťažší než ten istý problém bez precedenčnej relácie. Na riešenie tejto úlohy môžeme použiť v podstate ten istý postup vetiev a hraníc, ako bol práve popísaný s tým, že pri vetvení naviac kontrolujeme, či pridávaná úloha nenarušuje precedenčnú reláciu, čo je z enumeračného hľadiska výhodné, lebo ešte viac obmedzuje vetvenie.

2.3.2 Modely s paralelnými strojmi

Majme danú množinu $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$ m paralelných strojov, na ktorých sa má spracovať n úloh z množiny $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$. Každá úloha



Obr. 2.6: Ak by sme zaradili na k-te miesto v rozvrhu úlohu J_i, ktorá príde do systému v čase K^* alebo neskôr (t. j. $r_i \geq K^*$), vznikne nevyužitá medzera D dostatočne dlhá na to, aby sa v nej mohla spracovávať úloha J^*

J_i pozostáva z jedinej operácie $J_i = \{o_i\}$. Každá operácia o_i sa môže vykonať na ľubovoľnom stroji M_j s dobou spracovania p_{ij}. Ak na časy spracovania nie sú kladené ďalšie predpoklady, pôjde o najvšeobecnejší prípad $Rm|\beta|\gamma$ (*paralelné rôzne stroje*); ak sa dá písať $p_{ij} = p_i.q_j$, pôjde o systémy $Qm|\beta|\gamma$ (*paralelné uniformné stroje*), kde sa stroje líšia len rýchlosťou. Nakoniec ak doba spracovania operácie o_i je na všetkých strojoch rovnaká, t. j. ak $p_{ij} = p_i$, pôjde o systémy $Pm|\beta|\gamma$ (*paralelné identické stroje*).

Paralelné identické stroje

Ďalej sa budeme zaoberať prípadom $Pm|\beta|f$, kde f je regulárne kritérium. Označme p_i čas spracovania úlohy J_i (je rovnaký na každom stroji). Úlohy môžu byť nezávislé v tom zmysle, že nezáleží na poradí ich vykonávania, alebo na množine úloh $\mathcal{J}$ môže byť definovaná precedenčná relácia $\prec$. Množina rozvrhov bez prerušenia tu nie je dominantnou množinou ani pre regulárne kritériá, preto sa povoľuje v niektorých modeloch prerušenie vykonávania úloh. Budeme predpokladať, že všetky úlohy prídu do systému naraz v čase 0.

Veta 2.12. *V systémoch* $Pm||C_{\max}$ $Pm|pmtn|C_{\max}$ *je dĺžka najkratšieho rozvrhu väčšia alebo rovná*

$$L = \max\left\{\frac{1}{m}\sum_{i=1}^{n} p_i, \max_i\{p_i\}\right\} \tag{2.21}$$

Dôkaz. Keďže jedna úloha sa nemôže naraz spracovávať na dvoch strojoch, dĺžka najkratšieho rozvrhu musí byť aspoň $\max_i p_i$. Pretože stroj môže v jednom čase spracovávať len jednu operáciu, musí byť L väčšie alebo rovné priemernej dobe obsadenia jedného stroja. ■

Ukázali sme že v systémoch $Pm||C_{\max}$, $Pm|pmtn|C_{\max}$

$$C_{\max} \geq L = \max\left\{\frac{1}{m}\sum_{i=1}^{n} p_i, \max_i p_i\right\}$$

Skutočnosť, že v systéme $Pm|pmtn|C_{\max}$ už existuje prípustný rozvrh taký, že pre $C_{\max} = L$, ukáže nasledujúci algoritmus.

Algoritmus 2.5. McNaughtonov algoritmus pre systém $Pm|pmtn|C_{\max}$.

Postupne priraďuj úlohy tomu istému stroju, pokiaľ neprekročíš čas L z (2.21). Vykonávanie časti úlohy, ktorá by prekročila čas L, preruš a zaraď jej pokračovanie ďalšiemu stroju od času 0 atď., pokiaľ nepriradíš týmto spôsobom všetky úlohy.

Všeobecné paralelné stroje – systém $Rm||C_{\max}$

Nech p_{ij} je doba spracovania operácie o_i na stroji M_j. Označme x_{ij} pre $i = \{1, 2, \ldots, n\}$, $j = \{1, 2, \ldots, m\}$, binárnu premennú, ktorá nadobúda hodnotu 1 práve vtedy, keď úloha J_i je priradená stroju M_j. Označme C trvanie rozvrhu (t. j. čas ukončenia poslednej úlohy za predpokladu vstupu úloh v čase 0). Potom nájsť optimálny rozvrh v systéme $Rm||F_{\max}$ bez preempcie znamená riešiť nasledujúcu úlohu celočíselného programovania:

$$\text{Minimalizovať} \qquad C$$

za predpokladov

$$C - \sum_{i=1}^{n} p_{ij}.x_{ij} \geq 0 \qquad \text{pre} \quad j \in \{1, 2, \ldots, m\}$$

$$\sum_{j=1}^{m} x_{ij} = 1 \qquad \text{pre} \quad i \in \{1, 2, \ldots, n\}$$

$$x_{ij} \in \{0, 1\}$$

Takto definovanú úlohu je možné riešiť metódami celočíselného programovania. Je to však úloha NP–ťažká.

LPT algoritmus – heuristika pre $Pm||C_{\max}$

Nájsť optimálny rozvrh pre $Pm||C_{\max}$ je NP-ťažká úloha. Pre tento systém máme nasledujúci heuristický algoritmus, ktorý priraďuje strojom úlohy v poradí podľa najdlhšieho času spracovania (odtiaľ jeho názov LPT – Largest Processing Time).

Algoritmus 2.6. LPT – algoritmus
Krok 1: Vytvor poradie úloh $p_{[1]} \geq p_{[2]} \geq \cdots \geq p_{[n]}$.

Krok 2: V tomto poradí priraď úlohy strojom, ktoré sa najskôr uvoľnia.

Hlavnou myšlienkou LPT algoritmu je zaraďovať najprv úlohy s dlhou dobou spracovania tak, aby zaťaženie strojov bolo čo najrovnomernejšie s nádejou, že úlohy s krátkou dobou spracovania zaraďované nakoniec príliš nezhoršia kriteriálnu funkciu rozvrhu. Ukazuje sa, že tento jednoduchý algoritmus dáva veľmi dobré výsledky v praxi. Na rozdiel od mnohých heuristík, máme pre LPT algoritmus zaručené, že jeho riešenie sa neodchýli od optimálneho riešenia viac než cca o 33,3%, ako hovorí nasledujúca veta.

Veta 2.13. *Majme systém $Pm||C_{\max}$. Označme dĺžku optimálneho rozvrhu a dĺžku rozvrhu získaného algoritmom LPT porade $C_{\max}(OPT)$, $C_{\max}(LPT)$. Potom platí:*

$$\frac{C_{\max}(LPT)}{C_{\max}(OPT)} \leq \frac{4}{3} - \frac{1}{3m}$$

Algoritmus pre model $Pm||\overline{F}$ bez preempcie

Pre tento model máme algoritmus, ktorý zaraďuje úlohy podľa najväčšieho času spracovania (Greatest Processing Time).

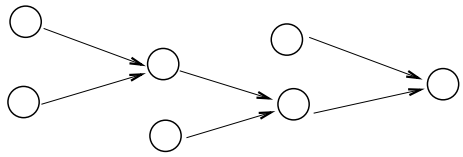
Algoritmus 2.7. GPT – algoritmus
Krok 1: Vytvor poradie úloh $p_{[1]} \leq p_{[2]} \leq \cdots \leq p_{[n]}$.

Krok 2: V tomto poradí postupne priraďuj úlohy strojom, ktoré sa najskôr uvoľnia.

Algoritmus dáva optimálny rozvrh. Možno ho použiť aj pre modely s preempciou, pretože v tomto prípade tvoria rozvrhy bez prerušenia dominantnú množinu.

Model $Pm|prec|C_{\max}$s precedenčnou reláciou bez preempcie

Zavedením precedenčnej relácie na množine úloh $\mathcal{J}$ sa väčšina úloh stáva NP-ťažkou. Sú však niektoré jednoduché prípady, pre ktoré existuje polynomiálny algoritmus. Jedným z nich je problém $Pm|p_i = 1,\ intree|F_{\max}$, v ktorom má každá úloha jednotkový čas spracovania, t. j. $p_i = 1$ pre $i = 1, 2, \ldots, n$ a precedenčná relácia má tvar montážneho stromu – každý uzol má najviac jedného následníka.



Obr. 2.7: Relácia bezprostrednej precedencie v tvare montážneho stromu

Huov algoritmus pre systém $Pm|p_i = 1,\ intree|F_{\max}$

Algoritmus 2.8. Huov algoritmus

Krok 1: Terminálnym úlohám (bez následníkov) daj značku a_i nulovú, ostatným priraď značku podľa úrovne (dĺžky najdlhšej orientovanej cesty od daného prvku ku niektorému terminálnemu prvku). V priebehu výpočtu bude $\mathcal{I}$ označovať množinu nezaradených úloh, $\mathcal{I}_0$ množinu tých úloh z $\mathcal{I}$, ktoré v $\mathcal{I}$ nemajú predchodcov. Ďalej označ $\langle t, t+1)$ časový interval, do ktorého budeme v kroku 2 zaraďovať úlohy.
Inicalizačne polož $t := 0$, $\mathcal{I} := \mathcal{J}$, $\mathcal{I}_0 \subseteq \mathcal{J}$ množina úloh bez predchodcov.

Krok 2: Ak je $|\mathcal{I}_0| \leq m$ (kde m je počet strojov), priraď všetky úlohy z $\mathcal{I}_0$ strojom do časového intervalu $\langle t, t+1)$, ostatné stroje nechaj pre tento časový interval voľné. Ak je $|\mathcal{I}_0| > m$, priraď do časového intervalu $\langle t, t+1)$ m úloh s najväčšími značkami.

Krok 3: Zaradené úlohy vylúč z množiny nezaradených úloh $\mathcal{I}$, a pokiaľ $\mathcal{I} \neq \varnothing$, aktualizuj $\mathcal{I}_0$, polož $t := t + 1$ a choď na krok 2.

Paralelné uniformné stroje

Systémy s paralelnými uniformnými strojmi sú charakteristické tým, že čas spracovania p_{ij} úlohy J_i na stroji M_j sa dá napísať ako $p_{ij} = p_i.q_j$, kde q_j sa nazýva faktor rýchlosti stroja M_j. V literatúre sa niekedy používa i zápis $p_{ij} = \dfrac{p_i}{v_j}$, kde $v_j = \dfrac{1}{q_j}$ sa nazýva rýchlosť stroja M_j.

Keďže systémy s preempciou dávali jednoduchšie výsledky pre paralelné identické stroje, budeme sa najprv zaoberať systémami $Qm|pmtn|C_{\max}$.

Veta 2.14. *Majme systém $Qm|pmtn|C_{\max}$. Nech platí:*

$$p_1 \geq p_2 \geq, \ldots, \geq p_n, \qquad v_1 \geq v_2 \geq, \ldots, \geq v_m.$$

Potom

$$C_{\max} \geq \max\left\{\frac{p_1}{v_1}, \frac{p_1+p_2}{v_1+v_2}, \ldots, \frac{\sum_{i=1}^{m-1} p_i}{\sum_{j=1}^{m-1} v_j}, \frac{\sum_{i=1}^{n} p_i}{\sum_{j=1}^{m} v_j}\right\}.$$

Dôkaz. Spracovanie všetkých úloh musí byť aspoň také dlhé, ako spracovanie najväčšej úlohy na najrýchlejšom stroji, čo predstavuje prvý člen na maxima na pravej strane. Takisto $C_{\max}$ musí byť väčšie ako čas, potrebný na spracovanie dvoch najväčších úloh na dvoch najrýchlejších strojoch pracujúcich rovnaký čas, čo vyjadruje druhý člen maxima na prvej strane. Ďalšie členy maxima až na posledný sa určia tým istým spôsobom. Posledný člen maxima vyjadruje, že spracovanie celej dávky nemôže byť kratšie ako spracovanie všetkých úloh na všetkých strojoch bežiacich rovnaký čas. ■

Algoritmus 2.9. LRPT–FM algoritmus systém $Qm|pmtn|C_{\max}$.

Krok 1: V každom časovom okamihu t urči ρ_i – nespracovanú časť úlohy J_i. Zoraď úlohy tak, aby

$$\rho_{[1]} \geq \rho_{[2]} \geq \cdots \geq \rho_{[n]}. \tag{2.22}$$

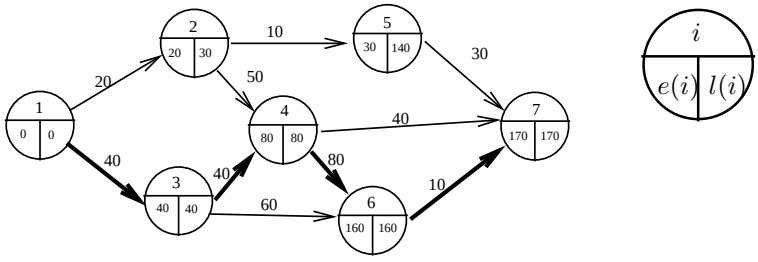
Krok 2: V tomto poradí priraď m úloh s najväčšími nespracovanými časťami strojom tak, že úlohu s najväčším nespracovaným zvyškom priradíš najrýchlejšiemu stroju, druhú úlohu v poradí LRPT priraď druhému najrýchlejšiemu stroju atď.

Na práve popísané pravidlo sa literatúra odvoláva ako na LRPT–FM (Longest Remaining Processing Time on Fastest Machine – najdlhší zostávajúci čas spracovania na najrýchlejšom stroji).

2.3.3 Metódy sieťového plánovania v kontexte teórie rozvrhov

Rozvrhovací problém, ktorý rieši metóda CPM, možno charakterizovať nasledujúco: Je daná množina úloh $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ a množina rovnakých paralelných strojov $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$. Každá z úloh J_i pozostáva z jedinej operácie o_i, ktorá sa vykoná na ľubovoľnom stroji M_j s časom spracovania p_i. Predpokladajme, že počet strojov m je dostatočne veľký na to, aby žiadna úloha pripravená na spracovanie nestála len preto, že pre ňu niet voľného stroja. (Na to stačí, aby $m \geq n$.) Na množine $\mathcal{J}$ je daná precedenčná relácia $\prec$. Úlohou je nájsť optimálny rozvrh z hľadiska minimalizácie kritéria $C_{\max}$ pri dodržaní relácie $\prec$. Hodnota $C_{\max}$ sa v kontexte sieťového plánovania naýva aj *trvanie projektu*.

Z hľadiska klasifikácie LLRK možno tento problém považovať za systém $P\infty|prec|C_{\max}$ s precedenčnou reláciou na množine úloh $\mathcal{J}$.



Obr. 2.8: Diagram sieťového digrafu, aký nájdete v mnohých učebniciach

V klasickej interpretácii sa nadväznosť úloh pre tento prípad modeluje tzv. *sieťovým digrafom*, v ktorom je každej úlohe J_i pridelená práve jedna hrana ohodnotená dĺžkou spracovania p_i úlohy J_i, vrcholy – začiatky a konce hrán predstavujú časové začiatky a konce spracovania úloh (reprezentovaných s nimi incidentnými hranami). Sieťový digraf obsahuje dva špeciálne vrcholy z – *začiatok vykonávania projektu* – jediný vrchol s nulovým vstupným stupňom a k – *koniec vykonávania projektu* – jediný vrchol s nulovým výstupným stupňom.

Sieťový digraf je neorientovane súvislý acyklický digraf, v ktorom pre každý vrchol x existuje z–x orientovaná cesta i x–k orientovaná cesta.

V acyklickom digrafe je možné očíslovať vrcholy tak, aby pre každú hranu (i, j) platilo $i < j$. Takéto očíslovanie budeme nazývať *monotónnym*. Jednou z metód monotónneho očíslovania acyklického digrafu G je nasledujúci postup:

Algoritmus 2.10. Monotónne očíslovanie acyklického digrafu

Krok 0: Nech $G = (V, H)$. Polož $i := 1$.

Krok 1: Nájdi vrchol x digrafu G, do ktorého nevedie žiadna hrana.
Priraď číslo i vrcholu x.

Krok 2: Ak $V - \{x\} = \emptyset$, STOP.
Inak polož $i := i + 1$, $G := G - \{x\}$ (odstráň vrchol x spolu s priľahlými hranami z digrafu G) a GOTO Krok 1.

Pri monotónnom očíslovaní vrcholov je $z = 1$, $k = N$, kde N je počet vrcholov sieťového digrafu.

Hodnotu $T = C_{\max}$ trvania projektu určíme ako

$$T = d_{\max}(z, k),$$

kde $d_{\max}(z, k)$ je dĺžka najdlhšej orientovanej cesty od začiatku projektu z do konca projektu k. Každú orientovanú cestu dĺžky T v sieťovom digrafe nazveme *kritickou cestou* (kritických ciest môže existovať viac). Úlohy (t. j. hrany) ležiace na kritickej ceste sa nazývajú *kritické úlohy*.

Pre každý vrchol x sieťového grafu určíme dva časové okamihy – $e(x)$ *najskôr možný začiatok činností* vychádzajúcich z vrchola x ako

$$e(x) = d_{\max}(z, x)$$

a $l(x)$ *najneskôr nutný koniec činností* vchádzajúcich do vrchola x ako

$$l(x) = T - d_{\max}(x, k),$$

kde $d_{\max}(x, y)$ je dĺžka najdlhšej x–y cesty.
Pre vrchol x ležiaci na nejakej kritickej ceste je $e(x) = l(x)$, pre ostatné vrcholy je $e(x) < l(x)$.

Úlohy reprezentované hranami (x, y) môžu byť v rozvrhu zadelené viacerými spôsobmi tak, že začínajú v ľubovoľnom časovom okamihu uzavretého intervalu $\langle e(x), l(y) - p_{xy}\rangle$, kde p_{xy} je doba spracovania úlohy reprezentovanej hranou (x, y).

Toto zadelenie je však obmedzené časovou polohou ostatných úloh v rozvrhu. Pre ľubovoľnú úlohu reprezentovanú hranou (x, y) a pre ľubovoľné $t \in \langle e(x), l(y) - p_{xy}\rangle$ existuje optimálny rozvrh taký, že daná úloha začína v čase t, ak však už raz fixujeme časovú polohu úlohy (x, y), pre ostatné úlohy tým obmedzíme ich časové polohy.

Zadelenie kritických úloh v čase je jednoznačné – ich začiatok je určený hodnotou $e(x) = l(x)$, kde x je vrchol sieťového digrafu reprezentujúci začiatok príslušnej úlohy.

Podľa teórie rozvrhov je rozvrh pre úlohy $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ určený postupnosťou časových intervalov $\{(b_1, c_1), (b_2, c_2), \ldots, (b_n, c_n)\}$, v ktorých sa majú úlohy z $\mathcal{J}$ vykonávať. Riešeniu úlohy sieťového plánovania ešte nedáva rozvrh v tomto zmysle, pretože, podľa predchádzajúceho textu, riešeniu zodpovedá niekoľko optimálnych rozvrhov.

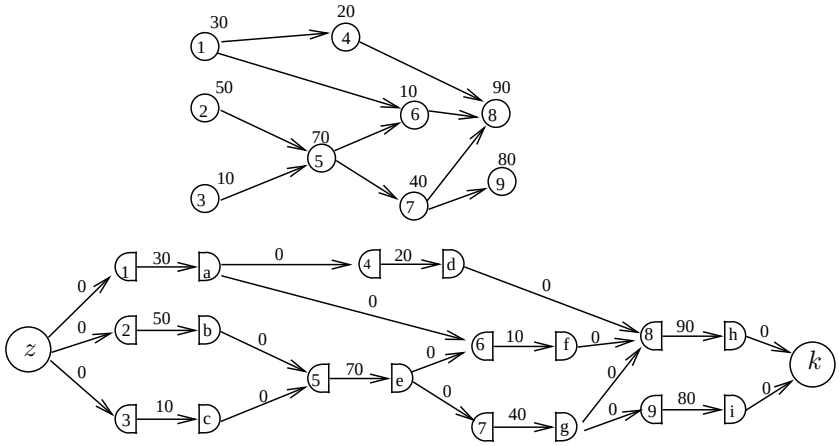
Jedným z nich je *ES–rozvrh* (early start schedule), v ktorom je štart každej z úloh naplánovaný v najskoršom možnom okamihu vypočítanom metódou CPM, iným je *LS–rozvrh* (late start schedule), v ktorom je štart každej úlohy naplánovaný tak, aby skončila v najneskôr nutnom čase určenom metódou CPM.

Výhodou metódy CPM je, že sa hodnoty najskôr možných začiatkov a najneskôr nutných koncov dajú vypočítať pomocou dĺžok najdlhších orientovaných ciest v acyklickom digrafe, (kde sa dá použiť napr. základný algoritmus so zápornou cenou hrany). Pre metódu CPM sa tak dajú použiť štandardné algoritmy na hľadanie extremálnych ciest v acyklickom digrafe. Touto výhodou sa však výpočet výhod končí.

Potiaže pri praktickej aplikácii CPM metódy začnú hneď pri pokuse vytvoriť k danej úlohe príslušný sieťový digraf. Jeden z postupov je vytvoriť najprv digraf $G_{\prec\prec} = (V, H)$ bezprostrednej precedencie a potom z neho vytvoriť príslušný sieťový digraf takto:

Každý vrchol množiny $v \in V$ rozdeliť na vstupnú v_I a výstupnú v_O časť a naviac do množiny vrcholov sieťového digrafu dodať dva špeciálne vrcholy z, k – začiatok a koniec projektu.

Do množiny hrán sieťového digrafu vložiť všetky orientované hrany spájajúce vstupnú a výstupnú časť pôvodného vrcholu, túto orientovanú hranu prehlásiť za činnosť a dať jej ohodnotenie – dĺžku činnosti. Pôvodné hrany digrafu $G_{\prec\prec}$ prehlásiť za fiktívne činnosti s nulovou dĺžkou trvania. Nakoniec dodať orientované hrany - fiktívne činnosti s nulovou dĺžkou trvania spájajúce začiatok z so všetkými vstupnými časťami vrcholov z $G_{\prec\prec}$ bez predchodcov, a fiktívne hrany, ktoré spájajú všetky výstupné časti vrcholov bez následníkov s koncom k.



Obr. 2.9: Konštrukcia sieťového digrafu (dole)
z precedenčného digrafu $G_{\prec\prec}$ (hore)

Výsledný digraf G_S má veľké množstvo orientovaných hrán s nulovou cenou - sú to tzv. *fiktívne činnosti*. Okrem toho, že zabezpečujú správnu nadväznosť elementárnych činností, pre prax nič nehovoria a sú dodané len preto, aby sa učinilo zadosť požiadavkám na sieťový digraf. Sú síce známe postupy, ako znižovať počet fiktívnych hrán, avšak úloha formulovaná ako: „*K danému digrafu bezprostrednej precedencie nájsť príslušný sieťový digraf s minimálnym počtom fiktívnych hrán*“ sa ukázala byť NP-ťažká, zatiaľ čo algoritmy na riešenie všetkých otázok sieťového plánovania prezentované v predchádzajúcej časti majú polynomiálnu zložitosť.

Teória rozvrhov namiesto sieťového digrafu používa graf $G_{\prec\prec}$ bezprostrednej precedencie úloh, ktorý sa na rozdiel od sieťového digrafu ľahko konštruuje. Algoritmy na konštrukciu ES–rozvrhu a LS–rozvrhu sú veľmi jednoduché:

Algoritmus 2.11. Algoritmus na konštrukciu ES–rozvrhu

Krok 0: Nech $J_1, J_2, \ldots, J_n$ je monotónne očíslovaná postupnosť úloh vzhľadom na reláciu precedencie $\prec$. Označ symbolom $a_{\min}(i)$ najskôr možný začiatok úlohy J_i.

Krok 1: Postupne pre $i = 1, 2, \ldots, n$ urob:
Ak J_i nemá predchodcu, polož $a_{\min}(i) := 0$.

Inak stanov $a_{\min}(i)$ ako maximum koncov spracovania všetkých bezprostredných predchodcov úlohy J_i.

Krok 3: Polož T rovné maximu koncov všetkých úloh bez následníkov.

Algoritmus 2.12. Algoritmus na konštrukciu LS–rozvrhu

Krok 0: Nech $J_1, J_2, \ldots, J_n$ je monotónne očíslovaná postupnosť úloh vzhľadom na reláciu precedencie $\prec$. Označ symbolom $l_{\max}(i)$ najneskôr nutný koniec úlohy J_i.

Krok 1: Postupne pre $i = n, n-1, \ldots, 2, 1$ urob:
Ak J_i nemá následníka, polož $l_{\max}(i) := T$.
Inak stanov $l_{\max}(i)$ ako minimum začiatkov spracovania všetkých bezprostredných následníkov úlohy J_i.

Ak by sme zmiernili požiadavku na trvanie projektu a dovolili, aby sa mohla posledná úloha dokončiť v čase $D > T$, potom by sa najneskôr nutný koniec vykonávania činností vchádzajúcich do vrchola x počítal ako $l(x) = D - d_{\max}(x, k)$. Tým vzniknú rezervy aj pre úlohy, ktoré pôvodne ležali na kritickej ceste. Takýmto dodatočným zmiernením požiadaviek na rozvrh sa zväčší stupeň voľnosti rozvrhu, ktorý možno využiť na splnenie niektorých dodatočných obmedzení – napr. na splnenie požiadavky na obmedzené zdroje.

Lineárny model pre úlohu sieťového plánovania

Predpokladajme, že úlohy $J_1, J_2, \ldots, J_n$ (resp. vrcholy digrafu precedencie $G_{\prec\prec} = (V, H, p)$) sú monotónne očíslované – ak $J_i \prec\prec J_j$, potom platí $i < j$. Predpoklad monotónneho očíslovania úloh pre samotný model lineárneho programovania nie je podstatný, avšak ako ďalej uvidíme, monotónne očíslovanie umožňuje vyriešiť nasledujúci problém jednoduchou metódou bez použitia všeobecných algoritmov lineárneho programovania.
Označme:

- x_i - začiatok vykonávania úlohy J_i
- p_i - trvanie úlohy J_i
- $\mathcal{I}$ - množina úloh bez predchodcov
- $\mathcal{T}$ - množina úloh bez následníkov

Potom pre úlohu sieťového plánovania možno zostaviť nasledujúci matematický model lineárneho programovania:

$$\text{Minimalizovať } \sum_{i=1}^{n+1} x_i \tag{2.23}$$

za podmienok:

$$x_i = 0 \qquad \text{pre } \forall i \quad i \in \mathcal{I} \tag{2.24}$$

$$x_j \geq x_i + p_i \qquad \text{pre } \forall i, j \in V \text{ také, že } J_i \prec\prec J_j \tag{2.25}$$

$$x_{n+1} \geq x_i + p_i \qquad \text{pre } \forall i \quad i \in \mathcal{T} \tag{2.26}$$

Pre optimálne riešenie $x_1^*, x_2^*, \ldots, x_n^*, x_{n+1}^*$ platí: x_i^* je najskôr možný začiatok úlohy J_i pre $i = 1, 2, \ldots, n$, x_{n+1}^* je optimálna dĺžka projektu, ktorú označíme symbolom T, t. j. $T = x_{n+1}^*$.

Z monotónneho očíslovania úloh vyplýva, že v každej z podmienok (2.25), (2.26) index premennej na ľavej strane je väčší, než index premennej na jej pravej strane. Ak teda usporiadame podmienky (2.24), (2.25), (2.26) neklesajúco podľa indexu premennej na ľavej strane, premenná x_i závisí len od hodnôt x_j, kde $j < i$ určených v predchádzajúcich podmienkach podľa tohto usporiadania. Stačí teda za optimálne x_i položiť najmenšie také, ktoré vyhovuje podmienkam (2.25). Tým je daný algoritmus na postupný výpočet hodnôt optimálneho riešenia tejto úlohy bez použitia aparátu lineárneho programovania.

Najneskôr nutné konce úloh $y_1^*, y_2^*, \ldots, y_n^*$ sú optimálnym riešením úlohy

$$\text{Maximalizovať } \sum_{i=1}^{n} y_i$$

za podmienok:

$$y_i = T \qquad \text{pre } \forall i \in \mathcal{T} \tag{2.27}$$

$$y_i \leq y_j - p_j \qquad \text{pre } \forall i, j \in V \text{ také, že } J_i \prec\prec J_j \tag{2.28}$$

Na riešenie tejto úlohy možno použiť analogický postup ako pri hľadaní najskôr možných začiatkov úloh s využitím monotónneho očíslovania úloh s tým rozdielom, že podmienky (2.27), (2.28) usporiadame nerastúco podľa indexu premennej na ľavej strane.

Zníženie ceny projektu za jeho predĺženie

Pre praktické riešenie úlohy sieťového plánovania je lineárne programovanie zbytočne silným aparátom, jeho význam však oceníme v prípadoch, kedy je kriteriálna funkcia rozvrhu zložitejšia než (2.23). Príkladom môže byť nasledujúce zovšeobecnenie tejto úlohy.

Predpokladajme, že časy p_i vykonávania jednotlivých úloh nie sú konštanty, ale každý sa môže meniť v intervale $\langle p_i, P_i \rangle$. Zrýchlenie vykonávania úlohy však čosi stojí. Nech $f_i(t) = c_{0i} - c_i.t$ pre $t \in \langle p_i, P_i \rangle$ je funkcia vyjadrujúca náklady na spracovanie úlohy J_i za predpokladu, že toto spracovanie bude trvať čas t.

Nech T je minimálna možná doba trvania projektu s časmi spracovania úloh p_i vypočítaná niektorou z metód klasickej úlohy sieťového plánovania. Nech P_i je horná hranica prípustného trvania úlohy J_i, $t_i \in \langle p_i, P_i \rangle$ nový (možno) predĺžený čas spracovania úlohy J_i, ktorý chceme zistiť. Ďalej označme x_{n+1} čas ukončenia poslednej úlohy a $x_1, x_2, \ldots, x_n$ začiatky spracovania úloh $J_1, J_2, \ldots, J_n$.

Nech C je pokuta za oneskorenie projektu o jednotku času. Ako kriteriálnu funkciu vezmeme súčet nákladov na spracovanie všetkých úloh plus celkovú pokutu za oneskorenie. Kriteriálna funkcia bude teda:

$$C(x_{n+1} - T) + \sum_i (c_{0i} - c_i t_i),$$

Pretože CT a $\sum_i c_{0i}$ sú konštanty, možno toto kritérium nahradiť jednoduchším

$$Cx_{n+1} - \sum_i c_i t_i \tag{2.29}$$

Úloha nájsť optimálne trvania t_i úloh J_i a optimálny rozvrh a s kriteriálnou funkciou (2.29) bude mať tvar:

$$\text{Minimalizovať } Cx_{n+1} - \sum_{i=1}^{n} c_i t_i \tag{2.30}$$

za podmienok:

$$x_i = 0 \qquad \text{pre } \forall i \quad i \in \mathcal{I} \tag{2.31}$$

$$x_j \geq x_i + t_i \qquad \text{pre } \forall i, j \in V \text{ také, že } J_i \prec\prec J_j \tag{2.32}$$

$$x_{n+1} \geq x_i + t_i \qquad \text{pre } \forall i \quad i \in \mathcal{T} \tag{2.33}$$

$$p_i \leq t_i \leq P_i \qquad \text{pre } i = 1, 2, \ldots, n \tag{2.34}$$

2.3.4 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 klasifikácii podľa LLRK pôjde o systémy $Fm||f$.

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 časom spracovania.

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 $M_1, M_2, \ldots, M_m$ a sú dané doby spracovania p_{ij} operácií o_{ij}.

Za týchto okolností je rozvrh určený poradím spracovania úlohy na každom stroji a takýchto poradí je $(n!)^m$.

Veta 2.15. *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. Zámenou poradia spracovania 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 2.16. *V systémoch $Fm||C_{\max}$ 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 predposlednom stroji môže skončiť skôr, než sa začne jej spracovanie na poslednom stroji. Zámenou poradia spracovania 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 predposlednom stroji, čo môže viesť i k urýchleniu spracovania poslednej úlohy na poslednom stroji. ■

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

Johnsonov problém

Najjednoduchším prípadom systému $Fm||C_{\max}$ je systém $F2||C_{\max}$. Úloha minimalizovať $C_{\max}$ 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.

Veta 2.17. *(Johnsonovo pravidlo) V optimálnom rozvrhu pre systém $F2||C_{\max}$ úloha J_i predchádza úlohu J_j, ak*

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

Dôkaz tohto tvrdenia je pomerne rozsiahly. Pretože presahuje rámec tejto publikácie, neuvádzame ho.

Algoritmus 2.13. Johnsonov algoritmus

Krok 1: Označ $\mathcal{I}$ množinu nezaradených úloh. Inicializačne polož $\mathcal{I} = \mathcal{J}$.

Krok 2: Nájdi $\min_{J_i \in \mathcal{I}} \{p_{i1}, p_{i2}\}$. 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: Polož $\mathcal{I} = \mathcal{I} - \{J_j\}$. Ak $\mathcal{I} = \varnothing$ STOP, inak choď na Krok 2.

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

Veta 2.18. *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 $F3||C_{\max}$ *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 $Fm||C_{\max}$ je NP–ťažkou úlohou. Ako už bolo spomenuté, ide o vybratie optimálneho riešenia spomedzi $(n!)^{(m-2)}$ možností v prípade kritéria $C_{\max}$, 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).

Algoritmus 2.14. Palmerova heuristika pre flow shop $Fm||C_{\max}$

Krok 1: Pre $j = 1, 2, \ldots, n$ vypočítaj

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

Krok 2: Ako suboptimálny rozvrh vezmi rozvrh, pre ktorý je

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

Algoritmus 2.15. Guptova heuristika pre flow shop $Fm||C_{\max}$

Krok 1: Pre $j = 1, 2, \ldots, n$ vypočítaj

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

kde

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

Krok 2: Ako suboptimálny rozvrh vezmi rozvrh, pre ktorý je

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

Uvádza sa, že Guptova heuristika (algoritmus 2.15) je vo väčšine prípadov lepšia, než Palmerova (algoritmus 2.14).

Heuristika Campbell, Dudek, Smith pre flow shop $Fm||C_{\max}$

Táto heuristika má $m-1$ etáp. V každej etape vytvorí z pôvodného $Fm||C_{\max}$ problému dvojstrojový problém $F2||C_{\max}$, 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 $C_{\max}$. Po $m-1$ etapách tak dostaneme $m-1$ permutačných rozvrhov (medzi ktorými môžu byť niektoré i rovnaké), z ktorých vyberieme ten s najmenšou hodnotou $C_{\max}$.
V jednotlivých etapách budú hodnoty p'_{i1}, p'_{i2} takéto:

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

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

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

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

...

...

ETAPA $m-1$: $p'_{i1} = \sum_{j=1}^{m-1} p_{ij}, \qquad p'_{i2} = \sum_{j=2}^{m} p_{ij}.$

Okrem spomínaných heuristík možno použiť i všetky metódy pre hľadanie optimálneho rozvrhu pre úlohy typu Job Shop, pretože Flow Shop je špeciálnym prípadom Job Shopu.

2.3.5 Job Shop systémy

Je daná množina úloh

$$\mathcal{J} = \{J_1, J_2, \ldots, J_n\},$$

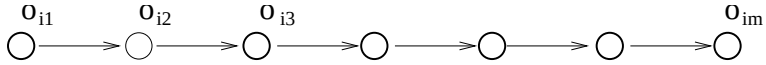
množina strojov

$$\mathcal{M} = \{M_1, M_2, \ldots, M_m\}.$$

Každá úloha J_i sa bude skladať z m operácií, t. j. $J_i = \{o_{i1}, o_{i2}, \ldots, o_{im}\}$, pričom pre každú úlohu J_i platí:

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

Graf bezprostrednej precedencie operácií úlohy J_i má veľmi jednoduchú štruktúru – má tvar lineárne usporiadanej retiazky – viď. obr. 2.10.



Obr. 2.10: Graf bezprostrednej precedencie pre operácie úlohy J_i

Graf bezprostrednej precedencie všetkých operácií všetkých úloh z $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ vznikne zjednotením takýchto lineárne usporiadaných retiazok.

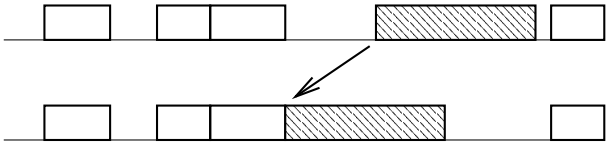
Na rozdiel od flow shop problémov, kde poradie prechodu strojmi bolo rovnaké pre všetky úlohy, v úlohách typu job shop je poradie prechodu rôznych úloh strojmi rôzne – pre každú operáciu o_{ij} ľubovoľnej úlohy J_i je priradený stroj $\mu_{ij} \in \mathcal{M}$, na ktorom sa táto operácia má vykonať s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

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 okamihu a na žiadnej úlohe sa nevykonávajú dve operácie naraz.

Uvedieme definície niekoľkých pojmov, ktoré sa často používajú v súvislosti s rozvrhovaním Job Shopov.

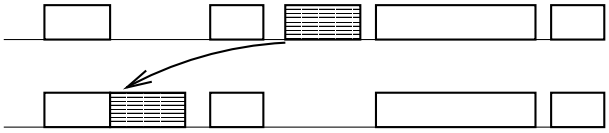
Lokálny ľavý posun (local left shift) v rozvrhu **S** je také časové posunutie začiatku spracovania niektorej operácie skôr, pri ktorom sa neporuší prípustnosť rozvrhu a zachová sa poradie spracovania operácií na každom stroji.

Globálny ľavý posun (global left shift) je taký presun niektorej operácie do skoršieho časového okamihu, pri ktorom sa zachová prípustnosť rozvrhu bez



Obr. 2.11: Lokálny ľavý posun

zmeny časovej polohy spracovania ostatných operácií.



Obr. 2.12: Globálny ľavý posun

Každý lokálny ľavý posun je aj globálnym ľavým posunom, nie však naopak – napríklad globálny ľavý posun z obrázka 2.12 nie je lokálnym ľavým posunom, lebo sa zmenilo poradie spracovania operácií.
Semiaktívny rozvrh je taký rozvrh, v ktorom neexistuje lokálny ľavý posun.
Aktívny rozvrh je taký rozvrh, v ktorom neexistuje globálny ľavý posun.

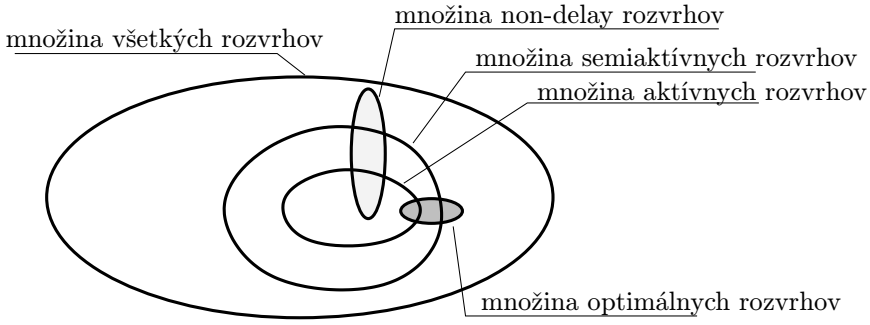
Majme semiaktívny rozvrh **S**. Podľa definície v ňom neexistuje lokálny ľavý posun, môže však v ňom existovať globálny ľavý posun. Rozvrh **S** nemusí byť aj aktívny.
Ak však máme aktívny rozvrh **S**, v ktorom podľa definície neexistuje globálny ľavý posun, potom je rozvrh **S** aj semiaktívny – pretože každý lokálny ľavý posun je aj globálnym ľavým posunom. Množina aktívnych rozvrhov je teda podmnožinou množiny semiaktívnych rozvrhov.

Umelý prestoj – delay 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 – non-delay schedule 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 je dané poradie úloh pre každý stroj, potom existuje jediný rozvrh, v ktorom nemožno urobiť ľavý lokálny posun.

Veta 2.19. *Množina semiaktívnych rozvrhov je dominantnou množinou pre regulárne optimalizačné kritérium.*
Množina aktívnych rozvrhov je dominantnou množinou pre regulárne optimalizačné kritérium.

Poznamenajme, že nie je záruka, že množina rozvrhov bez prestojov obsahuje optimálny rozvrh.



Obr. 2.13: Vzťah medzi množinami všetkých, aktívnych, semiaktívnych, non-delay a optimálnych rozvrhov

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

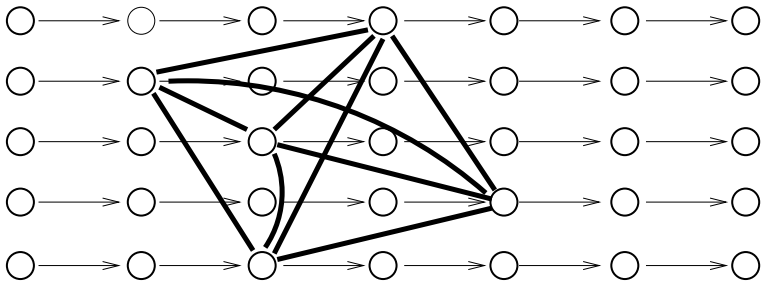
Zostrojme zmiešaný graf [2] $G = (V, H)$, ktorého množinou vrcholov je množina $\mathcal{O}$ všetkých operácií všetkých úloh z $\mathcal{J}$, t. j. $V = \mathcal{O}$, a ktorého hranovú množinu H tvoria hrany dvoch druhov – orientované a neorientované. Orientované hrany množiny H sú všetky usporiadané dvojice takých operácií, že druhá z nich sa musí vykonať bezprostredne po prvej. Sú to všetky dvojice operácií typu $(o_{ij}, o_{i(j+1)})$ – t. j. dve bezprostredne nasledujúce operácie jednej a tej istej úlohy J_i. Neorientované hrany množiny H sú všetky dvojice $\{o_{ij}, o_{kl}\}$ operácií súťažiace o ten istý stroj. Takéto hrany sa niekedy nazývajú aj *disjunktívne*

[2] Zmiešaný graf – migraf je taká štruktúra teórie grafov $G = (V, H)$, ktorej hranová množina H obsahuje orientované i neorientované hrany. Orientovanú hranu vychádzajúcu z vrchola u a vchádzajúcu do vrchola v modelujeme usporiadanou dvojicou (u, v), obyčajnú – neorientovanú hranu spájajúcu vrcholy u, v modelujeme neusporiadanou dvojicou – dvojprvkovou množinou $\{u, v\}$.

hrany.

$$H = \big\{(o_{ij}, o_{i(j+1)}) \mid i = 1, 2, \ldots, n, j = 1, 2, \ldots, m-1\big\} \tag{2.35}$$

$$\cup \big\{\{o_{ij}, o_{kl}\} \mid o_{ij},\ o_{kl} \text{ vyžadujú na spracovanie ten istý stroj}\big\} \tag{2.36}$$



Obr. 2.14: Model teórie grafov pre Job Shop

Na obrázku 2.14 orientované hrany modelujú bezprostrednú následnosť operácií v rámci jednej úlohy. Neorientovanými – disjunktívnymi – hranami sú pospájané operácie vyžadujúce rovnaký stroj. Pre prehľadnosť obrázok obsahuje neorientované hrany len pre jeden stroj. Vrcholy – operácie vyžadujúce ten istý stroj spolu s príslušnými neorientovanými hranami tvoria úplný podgraf multigrafu G s n vrcholmi. Zmiešaný graf G obsahuje práve m takýchto úplných podgrafov.

Úloha nájsť prípustný rozvrh pre systém $Jm||f$ je ekvivalentná úlohe prideliť disjunktívnym hranám zmiešaného grafu G také orientácie, aby takto vzniknutý graf $\overline{G}$ bol acyklický.

Ak už máme určené prípustné orientácie pre disjunktívne hrany, môžeme každú operáciu o_{ij} považovať za samostatnú úlohu $J_{ij} = \{o_{ij}\}$ o jedinej operácii. Označme $\overline{\mathcal{J}} = \{J_{ij} \mid i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, m\}$. Predpokladajme, že máme optimálny rozvrh $\mathbf{S}$ systému $P\infty| \prec |C_{\max}$ s precedenciou určenou grafom $\overline{G}$. Tento rozvrh určuje pre každú úlohu $J_{ij} = \{o_{ij}\}$ (a teda aj pre každú operáciu o_{ij}) časový interval spracovania (b_{ij}, c_{ij}). Pretože rozvrh $\mathbf{S}$ dodržuje precedenčnú reláciu danú orientáciou hrán grafu $\overline{G}$, musí platiť:

1. Žiadne dve operácie tej istej úlohy sa nevykonávajú v tom istom časovom okamihu.

2. Žiadne dve operácie vyžadujúce ten istý stroj sa nevykonávajú v tom istom časovom okamihu.

3. Intervaly spracovania operácií vyžadujúcich stroj $M_i \in \mathcal{M}$ sa dajú usporiadať do postupnosti disjunktných v čase lineárne usporiadaných časových intervalov, z čoho vyplýva, že sa tieto operácie dajú spracovať na stroji M_i.

Z uvedeného vyplýva, že každý rozvrh pre už definovanú úlohu $P\infty|prec|C_{\max}$ je prípustným rozvrhom aj pre pôvodnú úlohu $Jm||C_{\max}$.

Budeme hovoriť, že orientácia disjunktívnych hrán zmiešaného grafu G je *prípustná*, ak takto vzniknutý digraf $\overline{G}$ je acyklický. Úloha nájsť optimálny rozvrh pre systém $Jm||C_{\max}$ je teda ekvivalentná úlohe nájsť takú prípustnú orientáciu disjunktívnych hrán zmiešaného grafu G, pre ktorú je optimálne riešenie príslušného problému $P\infty|prec|C_{\max}$ minimálne.

Algoritmus na generovanie aktívnych rozvrhov

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 $i \in S_k$

$k_i = z_i + p_i$, 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.

Algoritmus 2.16. Giffler–Thompsonov algoritmus na generovanie aktívnych rozvrhov

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

Krok 2: Urči $k^* = \min_{i \in S_k}\{k_i\}$. Označ m^* stroj, ktorý vyžaduje operácia k^*.

Krok 3: Pre každú operáciu $i \in S_k$, ktorá vyžaduje stroj m^* a pre ktorú je $z_i < k^*$, vytvor 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 aktualizuj dáta nasledujúco:
a) Vylúč operáciu i z množiny S_k, t. j. $S_k := S_k - \{i\}$.
b) Vytvor S_{k+1} dodaním priameho následníka operácie i k S_k.
b) Inkrementuj k, t. j. $k := k + 1$.

Krok 5: GOTO Krok 2 pre každý čiastočný rozvrh $\mathcal{P}_k$ vytvorený v Kroku 3 a pokračuj týmto spôsobom, pokiaľ nevygeneruješ všetky aktívne rozvrhy.

Algoritmus na generovanie non-delay rozvrhov

Aktívnych rozvrhov je veľmi veľa. Preto sa niektoré prístupy obmedzujú na prehľadávanie semiaktívnych rozvrhov, ktoré však nemusia obsahovať optimálny rozvrh.

Ponechajme si označenie k, $\mathcal{P}_k$, z_i, k_i z algoritmu na generovanie aktívnych rozvrhov.

Algoritmus 2.17. Giffler–Thompsonov algoritmus na generovanie non-delay rozvrhov

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

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

Krok 3: Pre každú operáciu $i \in S_k$, ktorá vyžaduje stroj m^* a pre ktorú je $z_i = k^*$, vytvor 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 aktualizuj dáta nasledujúco:
a) Vylúč operáciu i z množiny S_k, t. j. $S_k := S_k - \{i\}$.
b) Vytvor S_{k+1} dodaním priameho následníka operácie i k S_k.
b) Inkrementuj k, t. j. $k := k + 1$.

Krok 5: GOTO Krok 2 pre každý čiastočný rozvrh $\mathcal{P}_k$ vytvorený v Kroku 3 a pokračuj týmto spôsobom, pokiaľ nevygeneruješ všetky non-delay rozvrhy.

Oba algoritmy navrhli Giffler a Thompson, preto sa na ne budeme ich menami v ďalšom odvolávať.

Počet non-delay 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 $z_i < k^*$.

Množinu operácií, o ktoré sa rozširuje parciálny rozvrh $\mathcal{P}_k$, nazvime *kritickou množinou* a označme $\mathcal{K}_k$. Vo väčšine prípadov množina $\mathcal{K}_k$ obsahuje veľa úloh, takže s rastúcim k počet parciálnych rozvrhov rastie exponenciálne, a preto sa algoritmy na generovanie aktívnych, resp. non-delay rozvrhov nezastavia v rozumnom čase. Je preto vhodné obmedziť vetvenie nejakými pravidlami, ktoré nazveme prioritnými pravidlami.

Jednou z možností je zobrať z kritickej množiny $\mathcal{K}_k$ len jednu operáciu a tou rozšíriť parciálny rozvrh $\mathcal{P}_k$.

SOT Shortest Operation Time - prednosť má operácia s najkratším časom spracovania

LOT Longest Operation Time - prednosť má operácia s najdlhším časom spracovania

SPT Shortest Processing Time - prednosť má operácia patriaca úlohe s najkratším celkovým časom spracovania

LPT Longest Processing Time - prednosť má operácia patriaca úlohe s najdlhším celkovým časom spracovania

SRPT Shortest Remaining Processing Time - prednosť má operácia patriaca úlohe s najkratším časom spracovania ešte nezaradených operácií

LRPT Longest Remaining Processing Time - prednosť má operácia patriaca úlohe s najdlhším časom spracovania ešte nezaradených operácií

RSRPT Relative Shortest Processing Time - prednosť má operácia o_{ij} úlohy J_i, ktorá má najmenšiu podielu $p_{ij}/RPT(i)$, kde p_{ij} je doba spracovania operácie o_{ij} a $RPT(i)$ je doba spracovania ešte nezaradených operácií úlohy J_i

RLRPT Relative Longest Processing Time - prednosť má operácia o_{ij} úlohy J_i, ktorá má najväčšiu hodnotu podielu $p_{ij}/RPT(i)$, kde p_{ij} je doba spracovania operácie o_{ij} a $RPT(i)$ je doba spracovania ešte nezaradených operácií úlohy J_i

Algoritmy založené na rozširovaní parciálneho rozvrhu podľa prioritného pravidla sú veľmi jednoduché a priamočiare, preto sme navrhli niekoľko ich rozšírení. Ako najperspektívnejšie sa ukázalo mnohonásobne spúšťať algoritmus s náhodným výberom operácie z $\mathcal{K}_k$, avšak pravdepodobnosti výberu operácií upraviť podľa hodnoty kriteriálnej funkcie príslušného prioritného pravidla.

Navrhnuté prístupy boli vyskúšané na niektorých príkladoch vytvorených Taillardom a tiež na inštanciách vygenerovaných Fisherom a Thompsonom. Tieto príklady sú bežne prístupné na internete. Ukázalo sa, že aj keď množina non-delay rozvrhov nie je dominantnou množinou, Giffler-Thompsonov algoritmus pre non-delay rozvrhy dáva často lepšie výsledky ako algoritmus pre aktívne rozvrhy. Po mnohých experimentoch sa ako najlepšie modifikácie s náhodným výberom podľa prioritného pravidla ukázali algoritmy pre non-delay rozvrhy, ktoré používali pravidlá RLRPT a LRPT. Odchýlky od skutočného optima, resp. od najlepšieho dosiahnutého výsledku boli na väčšine príkladov menšie než 10%.

Tento práve popísaný prístup je relatívne jednoduchý a ľahko naprogramovateľný, a preto je veľmi výhodný na riešenie mnohých praktických situácií, kedy je potrebné získať suboptimálne riešenie rýchlo. Dá sa tiež predpokladať, že rozvrhovacie inštancie z reálneho života nebudú obsahovať tak rafinované „pasce“, ako s tým zámerom umelo vytvorené inštancie, a teda suboptimálne riešenie nebude ďaleko od optima.

Ako doteraz najlepší prístup k riešeniu úloh typu Job Shop sa ukázalo použitie metaheuristiky Tabu Search, pri ktorej sa odchýlky od optima pohybovali pod tromi percentami.

„Shifting bottleneck“ heuristika

Táto heuristika štartuje z grafového modelu problému Job Shop (pozri obrázok 2.14 na str. 67), v ktorom sa bezprostredná následnosť operácií modeluje orientovanými hranami a súťaž operácií o ten istý stroj sa vyjadruje neorientovanými tzv. disjunktívnymi hranami.

Pri grafovej interpretácii Job Shop problému sme postupovali tak, že každú operáciu o_{ij} sme považovali za samostatnú úlohu $J_{ij} = \{o_{ij}\}$ o jedinej operácii. Ak na množine týchto operácií zavedieme precedenciu odvodenú od precedencie operácií v rámci pôvodných úloh, jej dodržanie zaručí, že žiadne dve operácie tej istej pôvodnej úlohy sa nebudú vykonávať súčasne. Ak určíme naviac prípustnú orientáciu disjunktívnych hrán – t. j. takú orientáciu, aby bol výsledný orientovaný graf acyklický, každý rozvrh, ktorý dodrží vzniknutú precedenciu operácií, bude prípustný i v zmysle pôvodnej formulácie úlohy Job Shop. Pre každú prí-

pustnú orientáciu disjunktívnych hrán tak dostávame problém $P\infty|prec|C_{\max}$. Nájsť optimálny rozvrh pre $Jm||C_{\max}$ sa tak prevedie na nájdenie takej prípustnej orientácie disjunktívnych hrán migrafu G, pre ktorú má optimálne riešenie príslušnej úlohy $P\infty|prec|C_{\max}$ najmenšiu hodnotu.

Postupovať budeme tak, že najprv budeme ignorovať všetky disjunktívne hrany. Označme G_0 graf, ktorý vznikne zo zmiešaného grafu G vypustením všetkých disjunktívnych hrán. V takomto probléme $P\infty|prec|C_{\max}$ určíme pre každú operáciu o_{ij} jej najskôr možný začiatok r_{ij} a najneskôr nutný koniec d_{ij} a trvanie T spracovania celej dávky úloh.

Vyberme ľubovoľný stroj M_k z množiny všetkých strojov $\mathcal{M}$. Nech $o_1, o_2, \ldots, o_n$ sú operácie, ktoré treba spracovať na stroji M_k, operáciu o_i s časom príchodu r_i a požadovaným časom ukončenia d_i. Ak sa nám podarí zaradiť intervaly spracovania operácií o_i do časových okien (r_i, d_i), je dobre. Ak spracovanie jednej operácie o_i skončí v čase $c_i > d_i$, spôsobí to predĺženie doby spracovania T celej dávky úloh o čas $L_i = (c_i - d_i)$. Ak je oneskorených operácií viac, meškanie celkového času ukončenia bude $\max_i\{L_i\}$. Pre stroj M_k teda treba nájsť také poradie úloh, pre ktoré je kritérium $L_{\max}$ minimálne – treba riešiť úlohu $1|r_i|\ L_{\max}$.

Ak vyriešime takto formulovanú úlohu pre každý stroj z množiny $\mathcal{M}$, nájdeme stroj M_b, pre ktorý je príslušná hodnota kritéria $L_{\max}$ najväčšia. Tento stroj prehlásime za „úzke hrdlo“ – „bottleneck“ a podľa poradia optimálneho riešenia úlohy $1|r_i|\ L_{\max}$ zavedieme orientáciu disjunktívnych hrán zmiešaného grafu G príslušných stroju M_b. Tieto orientované hrany pridáme k hranovej množine orientovaného grafu G_0, čím dostaneme orientovaný graf G_1.

V ďalšom kroku vyriešime problém $P\infty|prec|C_{\max}$ s precedenciou danou digrafom G_1, odkiaľ pre každú operáciu o_{ij} dostaneme časové okno (r_{ij}, d_{ij}). Pre každý stroj s doteraz neurčenou orientáciou disjunktívnych hrán vyriešime problém $1|r_i|\ L_{\max}$ a nájdeme stroj, pre ktorý je príslušná hodnota kritéria $L_{\max}$ najväčšia. Disjunktívnym hranám príslušným tomuto stroju dáme orientáciu podľa poradia daného optimálnym riešením úlohy $1|r_i|\ L_{\max}$ a zostrojíme orientovaný graf G_2 pridaním týchto orientovaných hrán ku hranám grafu G_1.

Takto pokračujeme dovtedy, kým neurčíme orientáciu všetkým disjunktívnym hranám zmiešaného grafu G.

Praktické skúsenosti hovoria, že pri takto formulovanom algoritme sa môže dostať poradie pri riešení problému $1|r_i|\ L_{\max}$ do rozporu s poradím určeným orientáciou príslušného orientovaného grafu G_k. To sa prejaví cyklením alebo havarovaním programu pri riešení nasledujúceho problému $P\infty|prec|C_{\max}$ v G_{k+1} (podľa spôsobu realizácie algoritmu), a to preto, lebo v tom prípade by v orientovanom grafe G_{k+1} vznikol cyklus. Tejto komplikácii sa dá ľahko vyhnúť tak, že

namiesto $1|r_i|\ L_{\max}$ riešime $1|r_i, prec|\ L_{\max}$ s precedenciou úloh danou orientáciou hrán orientovaného grafu G_k. Na túto skutočnosť literatúra neupozorňuje.

2.4 Neštandardné rozvrhovacie úlohy

Doteraz študované modely rozvrhovacích problémov najčastejšie pracovali s kriteriálnou funkciou $C_{\max}$, v ostatných prípadoch s regulárnym kritériom $f(C_1, C_2, \ldots, C_n)$, pričom hľadali optimálny rozvrh – pre každú operáciu o_i priradenie časového intervalu (b_i, c_i) na spracovanie tejto operácie a niekedy aj stroja, na ktorom sa táto operácia mala spracovať. Existujú však praktické problémy s univerzálnymi strojmi, pri ktorých sú časy spracovania dané a rozvrhovací problém spočíva v priradení operácií strojom tak, aby sa minimalizovali diametrálne iné kritériá ako tie, ktoré sme študovali doteraz. Jedným z najdôležitejších kritérií je minimalizovať počet použitých strojov, iným minimalizovať nerovnomernosť vyťaženia strojov, iným minimalizovať náklady spojené s prestavovaním strojov. Praktické úlohy prinášajú dokonca veľmi zložité komplexné kriteriálne funkcie, do ktorých sa dajú zahrnúť celkové náklady na realizáciu rozvrhu a dokonca i také požiadavky, ako sú sociálne požiadavky a požiadavky bezpečnosti práce.

Systém $Pm|s_{ij}, r_i, b_i = r_i|m$

Majme danú množinu úloh $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$, z ktorých každá úloha J_i pozostáva z jedinej operácie $J_i = \{o_i\}$. Úloha J_i sa má spracovať v časovom intervale (b_i, c_i), t. j. jej doba spracovania je $p_i = (c_i - b_i)$. Ak sa má na tom istom stroji spracovať úloha J_j bezprostredne po úlohe J_i, musí sa tento stroj prestavovať s_{ij} časových jednotiek.

Na spracovanie množiny úloh $\mathcal{J}$ možno použiť identické paralelné stroje. Úlohou je priradiť úlohy strojom tak, aby bolo v rozvrhu použitých čo najmenej strojov.

Z praktického významu prestavovacích časov s_{ij} vyplýva, že platí:

$$s_{ik} \leq s_{ij} + s_{jk}.$$

Doba s_{ik} je doba prestavenia stroja medzi úlohami J_i a J_k. Jedným zo spôsobov, ako to urobiť, je najprv prestaviť stroj z úlohy J_i na úlohu J_j a potom z úlohy J_j na úlohu J_k, čo trvá $s_{ij} + s_{jk}$ časových jednotiek. Doba s_{ik} priameho prestavenia stroja medzi úlohami J_i a J_k teda nemôže byť väčšia.

Na množine úloh $\mathcal{J}$ môžeme definovať reláciu $\prec$ predpisom

$$J_i \prec J_j \quad \text{práve vtedy, keď } c_i + s_{ij} \leq b_j. \tag{2.37}$$

Ak neplatí $J_i \prec J_j$, budeme písať $J_i \not\prec J_j$.
Z definície relácie $\prec$ ihneď vyplýva, že pre každú úlohu J_i platí $J_i \not\prec J_i$, lebo $b_i < c_i$. Ďalej vidieť, že nemôže platiť súčasne aj $J_i \prec J_j$ aj $J_j \prec J_i$.
Nech $J_i \prec J_j$, t. j. $c_i + s_{ij} \leq b_j$ a $J_j \prec J_k$, t. j. $c_j + s_{jk} \leq b_k$. Potom

$$c_i + s_{ik} \leq c_i + s_{ij} + s_{jk} \leq b_j + s_{jk} \leq c_j + s_{jk} \leq b_k,$$

a teda $J_i \prec J_k$. Relácia $\prec$ je tranzitívna a je reláciou precedencie na množine úloh $\mathcal{J}$.

Dve úlohy J_i, J_j sa môžu spracovať na tom istom stroji práve vtedy, ak $J_i \prec J_j$, alebo $J_j \prec J_i$. Budeme hovoriť, že úlohy J_i, J_j sú kolízne, ak neplatí ani $J_i \prec J_j$, ani $J_j \prec J_i$. Kolízne úlohy nemožno priradiť jednému stroju. Zostrojme graf $G_K = (V, H)$, s množinou vrcholov V rovnou množine všetkých úloh $\mathcal{J}$ a množinou hrán definovanou

$$H = \left\{ \{J_i, J_j\} \mid J_i \not\prec J_j,\ J_j \not\prec J_i \right\}$$

V prípustnom rozvrhu pre systém $Pm|s_{ij}, r_i, b_i = r_i|m$ je množina úloh priradená jednému stroju množinou takých vrcholov grafu G_K, z ktorých žiadne dva nie sú susedné (nie sú „spojené hranou“). Z algoritmickej teórie grafov je známa *úloha o farbení grafu*: Zafarbiť vrcholy grafu minimálnym počtom farieb tak, aby žiadne dva susedné vrcholy neboli zafarbené tou istou farbou. Formulácia tejto úlohy v reči výrobných rozvrhov je: Priradiť úlohám minimálny možný počet strojov tak, aby žiadnym dvom kolíznym úlohám nebol priradený ten istý stroj. Je známe, že úloha o farbení grafov je NP-ťažká. V našom prípade však ukážeme, že relácia $\prec$ na množine úloh $\mathcal{J}$ spôsobuje, že naša inštancia farbiacej úlohy je polynomiálne riešiteľná.

Definujme premennú x_{ij} nasledujúco

$$x_{ij} = \begin{cases} 1 & \text{ak sa má úloha } J_j \text{ spracovať bezprostredne} \\ & \text{za úlohou } J_i \text{ na tom istom stroji} \\ 0 & \text{inak} \end{cases} \tag{2.38}$$

Definujme čísla c_{ij} nasledujúco

$$c_{ij} = \begin{cases} 0 & \text{ak } J_i \prec J_j \\ 1 & \text{ak } J_i \not\prec J_j \end{cases}$$

Predstavme si, že už máme každej úlohe J_i určeného práve jedného následníka J_j. Ak $J_i \prec J_j$, t. j. ak $c_{ij} = 0$, môžu sa obe úlohy spracovať na tom istom stroji. Ak $J_i \not\prec J_j$, t. j. ak $c_{ij} = 1$, treba pre úlohu J_j ďalší stroj. Počet strojov pre priradenie dané maticou $\mathbf{X} = \left\{x_{ij}\right\}$ možno vyjadriť ako

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{2.39}$$

podmienky na priradenie sú známe podmienky priraďovacej úlohy

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \text{pre i=1,2,...,n} \tag{2.40}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \text{pre j=1,2,...,n} \tag{2.41}$$

$$x_{ij} \in \{0,1\} \quad \text{pre i,j=1,2,...,n} \tag{2.42}$$

Určiť minimálny potrebný počet strojov pre systém $Pm|s_{ij}, r_i, b_i = r_i|m$ znamenná určiť minimálnu hodnotu kriteriálnej funkcie (2.39) za podmienok (2.40), (2.41), (2.42), z optimálneho riešenia $\mathbf{X} = \left\{x_{ij}\right\}$ tejto úlohy dostaneme priradenie strojom nasledujúco:

Algoritmus 2.18. Algoritmus pre systém $Pm|s_{ij}, r_i, b_i = r_i|m$

Krok 0: Označ $\mathcal{I}$ množinu nezaradených úloh. Inicializačne $\mathcal{I} := \mathcal{J}$.
Označ m číslo aktuálneho stroja. Inicializačne $m := 0$.

Krok 2: Vyber $J_i \in \mathcal{I}$ také, že nemá v $\mathcal{I}$ predchodcu (v zmysle precedencie $\prec$).
Polož $m := m + 1$, priraď úlohu J_i stroju M_m a polož $\mathcal{I} = \mathcal{I} - \{J_i\}$

Krok 3: Pokiaľ existuje j také, že $x_{ij} = 1$, urob

- polož $i := j$ také, že $x_{ij} = 1$ (existuje len jedno také)
- priraď úlohu J_i stroju M_m
- polož $\mathcal{I} := \mathcal{I} - \{J_i\}$

Krok 4: Ak $\mathcal{I} = \varnothing$, STOP.
Inak GOTO Krok 2.

Rozvrhovacia úloha $Pm|s_{ij}, r_i, b_i = r_i|m$ má aj grafovú interpretáciu. Zostrojme precedenčný orientovaný grafu $G_\prec = (V, H)$, ktorého množinou vrcholov je množina $\mathcal{J}$ všetkých úloh a množinou orientovaných hrán je množina $H = \{(J_i, J_j)|\ J_i \prec J_j\}$. Orientovaný graf $G_\prec$ je tranzitívny acyklický digraf. Ak sa úlohy $J_{i_1}, J_{i_2}, \ldots, J_{i_k}$ dajú vykonať na jednom stroji, potom každé dve z nich sú porovnateľné v zmysle precedencie $\prec$, a preto sa dajú (po vhodnom prečíslovaní) usporiadať do postupnosti tvaru

$$J_{i_1} \prec J_{i_2} \prec \cdots \prec J_{i_k}. \tag{2.43}$$

Postupnosť (2.43) však zodpovedá orientovanej ceste v digrafe $G_\prec$. Naopak, všetky úlohy na každej orientovanej ceste v digrafe $G_\prec$ sa dajú priradiť jednému stroju. Minimalizovať počet strojov pre systém $Pm|s_{ij}, r_i, b_i = r_i|m$ znamená nájsť pokrytie všetkých vrcholov digrafu $G_\prec$ minimálnym počtom disjunktných orientovaných ciest.

Majme graf alebo digraf $G = (V, H)$. Budeme hovoriť, že množina $V_0 \subseteq V$ je nezávislá množina vrcholov grafu G, ak žiadne dva vrcholy množiny V_0 nie sú susedné (nie sú „spojené hranou“).

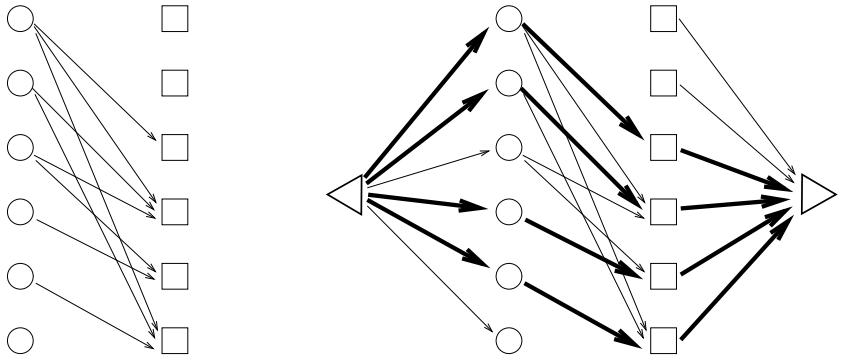
Veta 2.20. *DILWORTH. Nech G je acyklický tranzitívny digraf. Potom minimum počtu disjunktných ciest pokrývajúcich všetky vrcholy digrafu G sa rovná maximu mohutností všetkých nezávislých množín vrcholov grafu G.*

Existuje ešte jedna grafová reprezetácia študovanej úlohy. Zostrojme digraf $G_B = (V, H)$, ktorý bude mať pre každú úlohu $J_i \in \mathcal{J}$ dva vrcholy – jeden c_i pre koniec a druhý b_i pre začiatok vykonávania úlohy J_i. Označme $V_1 = \{c_i \mid i = 1, 2, \ldots, n\}$, $V_2 = \{b_i \mid i = 1, 2, \ldots, n\}$. Potom $V = V_1 \cup V_2$. Množina hrán H digrafu $G_B = (V, H)$ bude definovaná nasledujúco

$$H = \{(c_i, b_j) \mid J_i \prec J_j\}.$$

Digraf G_B je bipartitný – dvojčasťový digraf s časťami V_1, V_2.

Predstavme si najprv, že pre každú úlohu máme jeden stroj – koľko úloh, toľko strojov. Ak chceme ušetriť jeden stroj, môžeme to urobiť tak, že realizujeme jednu z následností – vyberieme jednu hranu (c_i, b_j) z množiny H, čím povieme, že stroj, ktorý vykonáva úlohu J_i po jej vykonaní spracuje aj úlohu J_j. Takto môžem vybrať ďalšiu hranu c_k, b_l a postupne ďalšie a vybratím každej ušetriť jeden stroj. Vyberanie hrán však nemôže byť úplne ľubovoľné, vybrané hrany musia byť po dvoch disjunktné – nesmú mať rovnaký ani počiatočný ani koncový vrchol. Rovnaký počiatočný vrchol dvoch vybratých hrán napr. (c_i, b_j), (c_i, b_k) by znamenal, že stroj po spracovaní úlohy J_i spracuje obe úlohy J_j aj



Obr. 2.15: Bipartitný graf G_B a) a jeho doplnenie zdrojom a ústím na sieť spolu s prípustným tokom b)

J_k, čo by viedlo k súčasnému spracovaniu kolíznych úloh. Rovnosť koncových vrcholov dvoch vybratých hrán napr. (c_i, b_k), (c_j, b_k) by hovorila, že stroj, ktorý spracoval úlohu J_i a stroj, ktorý spracoval úlohu J_j oba následne spracujú úlohu J_k, čo by odporovalo pravidlu, že úloha je priradená len jednému stroju.

Úloha o minimalizácii počtu strojov vedie v bipartitnom digrafe G_B k nasledujúcej formulácii: V digrafe G_B nájsť najpočetnejšiu podmnožinu disjunktných hrán, čo je to isté ako nájsť v G_B najpočetnejšie párenie. Je známe, že takto formulovaná úloha sa rieši tak, že k množine vrcholov V digrafu G_B pridáme dva nové vrcholy - zdroj z a ústie u, množinu hrán H rozšírime o všetky orientované hrany typu (z, c_i), (b_i, u) pre $i = 1, 2 \ldots, n$, všetkým hranám priradíme kapacitu 1 a v takto vzniknutej sieti hľadáme maximálny tok. Ak hranou typu (b_i, c_j) tečie jednotkový tok, potom hrana patrí do vybranej množiny hrán – najpočetnejšieho párenia.

Systémy $Pm|s_{ij}, r_i, b_i = r_i|F$

Minimalizácia počtu strojov často nestačí praktickým požiadavkám na efektívnosť výrobného procesu. Priradení s rovnakým minimálnym počtom strojov býva často veľké množstvo a z týchto rozvrhov treba vybrať taký rozvrh $\mathbf{S}$, ktorý je z istého hľadiska najlepší. Kvalitu rozvrhu definuje kriteriálna funkcia $F(\mathbf{S})$.

Rozvrh $\mathbf{S}$ možno interpretovať viacerými spôsobmi – maticou $\mathbf{X} = \{x_{ij}\}$, priradením, t. j. permutáciou $\pi[i]$ takou, že $x_{i\pi[i]} = 1$, t. j. že úloha $J_{\pi[i]}$ sa spracuje bezprostredne za úlohou J_i na tom istom stroji atď.

Na účely klasifikácie kriteriálnych funkcií bude výhodná nasledujúca reprezentácia.

$$\begin{aligned}
S_M(1) =& (J_{[11]} \prec J_{[12]} \prec \cdots \prec J_{[1,k_1]}) \\
S_M(2) =& (J_{[21]} \prec J_{[22]} \prec \cdots \prec J_{[2,k_2]}) \\
& \cdots \\
S_M(i) =& (J_{[i1]} \prec J_{[i2]} \prec \cdots \prec J_{[i,k_i]}) \\
& \cdots \\
S_M(m) =& (J_{[m1]} \prec J_{[m2]} \prec \cdots \prec J_{[m,k_m]}),
\end{aligned}$$

kde $S_M(i)$ je postupnosť úloh, ktoré sa spracujú na stroji M_i.

Kriteriálnu funkciu $F(\mathbf{S})$ nazveme *separabilnou*, ak existuje taká funkcia f, že

$$F(\mathbf{S}) = \sum_{i=1}^{m} f\big(S_M(i)\big). \tag{2.44}$$

Kriteriálnu funkciu F nazveme *lineárnou*, ak je separabilná a funkcia f je v tvare

$$f\big(S_M(i)\big) = \sum_{j=1}^{k_i-1} c\big(J_{[ij]}, J_{[i(j+1)]}\big), \tag{2.45}$$

kde $c\big(J_i, J_j\big)$ predstavuje náklady na prestavenie stroja z úlohy J_i na úlohu J_j, ak $J_i \prec J_j$, inak $c\big(J_i, J_j\big) = \infty$.

Systémy $Pm|s_{ij}, r_i, b_i = r_i|F$ s lineárnou kriteriálnou funkciou F

Nech x_{ij} je premenná definovaná v (2.38) (str. 74), $\mathbf{S}$ rozvrh príslušný matici $\mathbf{X} = \{x_{ij}\}$. Nech F je lineárna kriteriálna funkcia splňujúca 2.44, 2.45. Pre skrátenie zápisu označme $c_{ij} = c(J_i, J_j)$. Potom platí:

$$F(\mathbf{S}) = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{2.46}$$

Ohraničujúce podmienky pre priraďovaciu úlohu sú

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \text{pre i=1,2,...,n} \tag{2.47}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \text{pre j=1,2,...,n} \tag{2.48}$$

$$x_{ij} \in \{0, 1\} \tag{2.49}$$

Nájsť optimálny rozvrh pre špecifikovaný systém znamená minimalizovať kriteriálnu funkciu (2.46) za predpokladov (2.47), (2.48) a (2.49). Takto formulovaná úloha je známa priraďovacia úloha, pre ktorú existujú polynomiálne algoritmy riešenia.

Systémy $Pm|s_{ij}, r_i, b_i = r_i|F$ so separabilnou nelineárnou kriteriálnou funkciou F

Ak prestáva platiť podmienka linearity (2.45), vznikajúca úloha je pravdepodobne NP-ťažká. Ak stále platí podmienka separability, existujú heuristické postupy, ktoré dávajú v praxi veľmi dobré výsledky.
Majme postupnosť úloh pre stroj J_i

$$S_M(i) = (J_{[i1]} \prec J_{[i2]} \prec \cdots \prec J_{[i,k_i]}).$$

Túto postupnosť môžeme rozdeliť na dve časti

$$S_M(i) = (\underbrace{J_{[i1]} \prec J_{[i2]} \prec \cdots \prec J_{[i,p]}}_{\text{Začiatok } Z(i)} \prec \underbrace{J_{[ip+1]} \prec J_{[ip+2]} \prec \cdots \prec J_{[i,k_i]}}_{\text{Koniec K(i)}}).$$

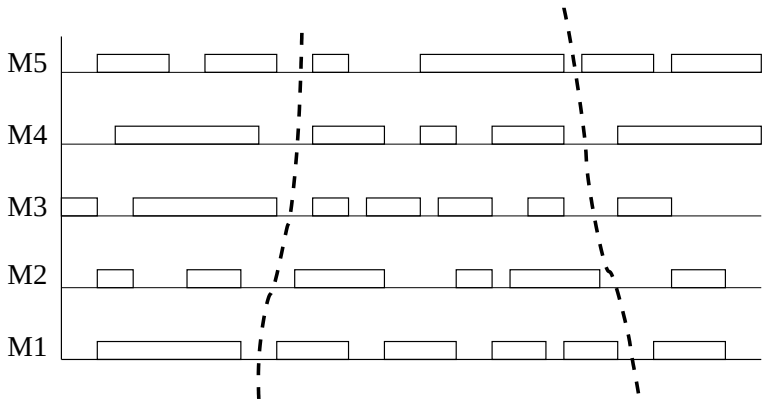
Označme $S_M(i,j)$ taký rozvrh pre stroj M_i, v ktorom ponecháme začiatok $Z(i)$ rozvrhu $S_M(i)$, ale jeho koniec nahradíme koncom $K(j)$ rozvrhu stroja $S_M(j)$. Označme

$$d_{ij} = \begin{cases} f\big(S_M(i,j)\big) & \text{ak } S_M(i,j) \text{ je prípustný rozvrh} \\ \infty & \text{inak} \end{cases}$$

Nech $\mathbf{Y} = \{y_{ij}\}$ pre $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, m$ je matica priradenia koncov $K(j)$ začiatkom $Z(i)$. Vznikne tak nový rozvrh $\overline{\mathbf{S}}$, ktorého cena sa vypočíta ako

$$F(\overline{\mathbf{S}}) = \sum_{i=1}^{m} \sum_{j=1}^{m} d_{ij} y_{ij}. \tag{2.50}$$

Optimálne priradiť začiatky a konce rozvrhov pre jednotlivé stroje znamená vyriešiť priraďovaciu úlohu s kriteriálnou funkciou (2.50).



Obr. 2.16: Rozdelenie rozvrhu na začiatky, stredy a konce

Je možný ešte detailnejší prístup, a to rozdeliť každý strojový rozvrh na tri časti: začiatok, stred a koniec, a definovať strojový rozvrh $S_M(i,j)$ ako ten rozvrh pre stroj M_i, ktorý dostaneme tak, že začiatok a koniec rozvrhu $S_M(i)$ ponecháme a jeho stred nahradíme stredom rozvrhu $S_M(j)$ stroja M_j. Delenie strojových rozvrhov na tri časti ilustruje obrázok 2.16.

Na základe uvedeného sme navrhli nasledujúci heuristický algoritmus, ktorý sa uplatnil v praxi s veľmi dobrými výsledkami.

Algoritmus 2.19. Heuristický algoritmus pre systém $Pm|s_{ij}, r_i, b_i = r_i|F$

Krok 1: Nájdi optimálny rozvrh $\mathbf{S}$ pre $Pm|s_{ij}, r_i, b_i = r_i|m$

Krok 2: Hľadaj také rozdelenie strojových rozvrhov určených rozvrhom $\mathbf{S}$ na začiatky a konce, (alebo začiatky, stredy a konce), pre ktoré dá priraďovacia úloha s kriteriálnou funkciou (2.50) rozvrh $\overline{\mathbf{S}}$ taký, že $F(\overline{\mathbf{S}}) < F(\mathbf{S})$.

Krok 3: Ak také rozdelenie existuje, polož $\mathbf{S} := \overline{\mathbf{S}}$ a GOTO Krok 2.

Krok 4: Ak také rozdelenie neexistuje, STOP, $\mathbf{S}$ je suboptimálny rozvrh.

Kapitola 3

Dopravná a distribučná logistika

3.1 Dopravná sieť

Základnými prvkami dopravnej siete sú *uzly* a *úseky dopravnej siete*. Uzly dopravnej siete sú pospájané úsekmi dopravnej siete. Uzly i úseky dopravnej siete môžu mať rôzne číselné charakteristiky. Najčastejšími charakteristikami úseku dopravnej siete sú dĺžka a priepustnosť úseku. S uzlom sa často spája jeho náročnosť na obsluhu, kapacita uzla (ak je uzol dodávateľom) alebo jeho požiadavky (ak je uzol odberateľom). Podľa charakteru riešeného problému uzlom i úsekom môžu byť priradené aj mnohé ďalšie číselné charakteristiky ako spoľahlivosť, náklady na údržbu, časové zdržanie pri prechode úsekom či uzlom a podobne.

Úseky dopravnej siete môžu byť prechádzané v oboch smeroch (obojsmerné úseky), alebo je dovolené prechádzať ich len v jednom smere.

Základným prostriedkom na modelovanie dopravnej siete je graf a digraf.

3.2 Extremálne cesty v dopravných sieťach

Významným problémom pre dopravnú logistiku je hľadanie optimálnej cesty v dopravnej sieti. Kritérium optimality môže byť rôzne. Najčastejšou požiadavkou je hľadanie najkratšej cesty – cesty s minimálnou dĺžkou.

3.2.1 Najkratšia cesta

Najkratšia orientovaná u–v cesta v digrafe $G = (V, H, c)$ je tá orientovaná u–v cesta, ktorá má najmenšiu dĺžku. *Najkratšia u–v cesta* v grafe je definovaná analogicky.

V tejto časti sa budeme venovať hľadaniu najkratšej cesty v digrafe. Pre neorientovaný graf sa problém prevedie na hľadanie cesty v orientovanom grafe – digrafe tak, že každá neorientovaná hrana $\{u, v\}$ sa nahradí dvojicou orientovaných hrán (u, v), (v, u) s rovnakou dĺžkou, ako mala hrana $\{u, v\}$.

Pre výpočet najkratšej cesty v digrafe existuje mnoho algoritmov. Uvedieme z nich dva - *základný algoritmus* a *Dijkstrov algoritmus*.

Algoritmus 3.1. Základný algoritmus na hľadanie najkratších u–v ciest z pevného vrchola $u \in V$ do všetkých ostatných vrcholov $v \in V$ v hranovo ohodnotenom digrafe $G = (V, H, c)$ s nezápornou cenou hrany $c(h)$.

Krok 1: Inicializácia. Pre každý vrchol $i \in V$ priraď dve značky $t(i)$ a $x(i)$. Značka $t(i)$ predstavuje horný odhad dĺžky doteraz nájdenej najlepšej u–i cesty a $x(i)$ je jej predposledný vrchol. Polož $t(u) = 0$, $t(i) = \infty$ pre $i \in V,\ i \neq u$ a $x(i) = 0$ pre každé $i \in V$.

Krok 2: Hľadaj orientovanú hranu $(i, j) \in H$, pre ktorú platí $t(j) > t(i) + c(i, j)$. Ak taká hrana existuje, potom polož $t(j) := t(i) + c(i, j)$, $x(j) := i$ a opakuj Krok 2.

Krok 3: Ak taká hrana neexistuje, potom $t(i)$ predstavuje dĺžku najkratšej u–i cesty pre každý vrchol i. Najkratšiu u–i cestu zostroj potom spätne pomocou značiek $x(i)$ ako cestu prechádzajúcu vrcholmi

$$i, x(i), x(x(i)), x(x(x(i))), \ldots, u,$$

teda táto najkratšia cesta bude mať tvar

$$u, \ldots, x(x(i))), \underbrace{(x(x(i))), x(x(i)))}_{\text{hrana}}, x(x(i)),$$
$$\underbrace{(x(x(i)), x(i))}_{\text{hrana}}, x(i), \underbrace{(x(i), i)}_{\text{hrana}}, i$$

Ako jeden z najefektívnejších uvádzame nasledujúci Dijkstrov algoritmus.

Algoritmus 3.2. Dijkstrov algoritmus na zostrojenie najkratšej u–v cesty v hranovo ohodnotenom digrafe $G = (V, H, c)$ s nezáporným ohodnotením hrán.

Krok 1: Inicializácia. Pre každý vrchol $i \in V$ priraď dve značky $t(i)$ a $x(i)$. Značky $t(i)$ budú dvojakého druhu, a to dočasné (ktoré sa ešte v priebehu výpočtu môžu zmeniť) a konečné (ktoré sa už nemôžu zmeniť).

Polož $t(u) = 0$, $t(i) = \infty$ pre $i \in V,\ i \neq u$ a $x(i) = 0$ pre každé $i \in V$. Zvoľ riadiaci vrchol $r := u$ a značku t pri vrchole $u = r$ prehlás za konečnú.

Krok 2: Ak je $r = v$, stop. Značka $t(v)$ predstavuje dĺžku najkratšej u–v cesty. Najkratšiu u–v cestu zostroj potom spätne pomocou značiek $x(i)$ ako cestu prechádzajúcu vrcholmi

$$v, x(v), x(x(v)), x(x(x(v))), \ldots, u,$$

teda táto najkratšia cesta bude mať tvar

$$u, \ldots, x(x(x(v))), \underbrace{(x(x(x(v))), x(x(v)))}_{\text{hrana}}, x(x(v)),$$
$$\underbrace{(x(x(v)), x(v))}_{\text{hrana}}, x(v), \underbrace{(x(v), v)}_{\text{hrana}}, v$$

Inak pre všetky hrany tvaru $(r, j) \in H$, kde j je vrchol s dočasnou značkou, urob:

Ak $t(j) > t(r) + c(r, j)$, potom $t(j) := t(r) + c(r, j)$, $x(j) := r$.

Krok 3: Zo všetkých dočasne označených vrcholov nájdi ten vrchol i, ktorý má značku $t(i)$ minimálnu. Značku pri tomto vrchole i prehlás za konečnú a zvoľ nový riadiaci vrchol $r := i$. (Pokiaľ existuje viac vrcholov s rovnakou minimálnou značkou, akú má vrchol i, za definitívnu značku môžeme prehlásiť len značku pri jednom z týchto vrcholov – ten potom berieme za riadiaci. Na tie ďalšie dôjde v nasledujúcich krokoch výpočtu). Choď na Krok 2.

Ak v kroku 2. nahradíme podmienku zastavenia algoritmu „Ak je $r = v$“ podmienkou „Ak sú všetky vrcholy označené definitívnymi značkami, alebo ak $t(r) = \infty$“, potom algoritmus vypočíta všetky najkratšie cesty u–i cesty, pričom $t(i)$ predstavuje dĺžku najkratšej u–i cesty pre každý vrchol i.

Pri oboch algoritmoch sa môže stať, že pre niektoré vrcholy i dostaneme $t(i) = \infty$. Do takýchto vrcholov nevedie žiadna orientovaná cesta z vrchola u.

Algoritmus 3.3. Label-set a Label-correct implementácia algoritmu na hľadanie najkratších u–v ciest z pevného vrchola $u \in V$ do všetkých ostatných

vrcholov $v \in V$ v hranovo ohodnotenom digrafe $G = (V, H, c)$ s nezápornou cenou hrany $c(h)$.

Krok 1: Inicializácia. Polož $t(u) := 0$, $t(i) := \infty$ pre $i \in V,\ i \neq u$ a $x(i) := 0$ pre každé $i \in V$. Polož $\mathcal{E} := \{u\}$.

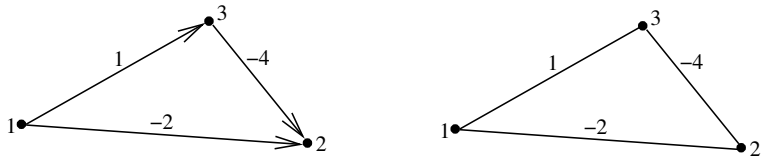
Krok 2: Vyber $r \in \mathcal{E}$, polož $\mathcal{E} := \mathcal{E} - \{r\}$.
Pre všetky hrany tvaru $(r, j) \in H$ urob:
Ak $t(j) > t(r) + c(r, j)$, potom $t(j) := t(r) + c(r, j)$, $x(j) := r$, $\mathcal{E} := \mathcal{E} \cup \{j\}$.

Krok 3: Ak $\mathcal{E} \neq \emptyset$, choď na Krok 2.
Ak $\mathcal{E} = \emptyset$, potom $t(i)$ predstavuje dĺžku najkratšej u–i cesty pre každý vrchol i. Najkratšiu u–i cestu zostroj potom spätne pomocou značiek $x(i)$ ako v predchádzajúcich dvoch algoritmoch.

Ak v druhom kroku algoritmu 3.3 vyberáme $r \in \mathcal{E}$ ľubovoľne, dostávame implementáciu základného algoritmu, ktorú voláme *label correct algoritmus*. Ak za prvok $r \in \mathcal{E}$ vyberáme prvok s najmenšou značkou $t()$, potom dostaneme implementáciu Dijkstrovho algoritmu, ktorú voláme *label set algoritmus*. Pre label correct algoritmus je výhodné organizovať $\mathcal{E}$ ako zásobník, pre label set algoritmus sa $\mathcal{E}$ organizuje ako prioritný front, prípadne ako halda.

3.2.2 Cyklus zápornej dĺžky v digrafe

Všetky spomínané algoritmy pre najkratšiu cestu sú rýchle. Dá sa ukázať, že ich zložitosť je polynomiálna, ak sa použijú pre digrafy s nezáporným ohodnoteniami hrán. Pre prípad všeobecného ohodnotenia hrán je však situácia zložitejšia.



Obr. 3.1: Digraf a graf so všeobecne ocenenými hranami, v ktorých Dijkstrov algoritmus zlyháva napriek neexistencii cyklu zápornej ceny, ak počítame vzdialenosti z vrchola 1

Hľadanie najkratšej cesty vo všeobecne ohodnotenom grafe je úloha polynomiálne ekvivalentná s úlohou obchodného cestujúceho – je to NP-ťažká úloha.

V prípade, že v digrafe existujú hrany zápornej dĺžky, Dijkstrov algoritmus nebude pracovať správne a ostatné spomínané algoritmy sa zacyklia (nikdy sa nezastavia) v prípade, že v digrafe existuje cyklus, ktorého dĺžka je záporná.

V prípade, že vo všeobecne ohodnotenom digrafe neexistuje cyklus zápornej dĺžky, úloha hľadania najkratšej cesty ostáva úlohou s polynomiálnou zložitosťou a ešte stále možno použiť základný algoritmus, resp. label-correct algoritmus na výpočet najkratších ciest.

Úlohu hľadania najdlhšej cesty v digrafe $G = (V, H, c)$ možno previesť na úlohu hľadania najkratšej cesty v digrafe $\overline{G} = (V, H, d)$, kde $d(h) = -c(h)$ pre všetky hrany $h \in H$. Ak digraf $\overline{G}$ neobsahuje cyklus zápornej dĺžky, je úloha hľadania najdlhšej cesty v pôvodnom digrafe G polynomiálne riešiteľná.

Nech teraz $G = (V, H, c)$ je všeobecne ohodnotený graf, nech $h = \{u, v\} \in H$ je hrana so zápornou cenou $c(h) < 0$. Ako už bolo spomenuté, najkratšiu cestu v grafe G hľadáme ako najkratšiu cestu v digrafe s rovnakou množinou vrcholov a množinou hrán, ktorá ku každej hrane $\{x, y\} \in H$ obsahuje dvojicu hrán (x, y), (y, x), obe s cenou rovnou $c\{x, y\}$. Ak však cena hrany $h = \{u, v\} \in H$ bola záporná, potom $u, (u, v), v, (v, u), u$ je cyklus zápornej dĺžky a preto sú algoritmy na hľadanie najkratšej cesty pre všeobecne ohodnotené grafy nepoužiteľné.

Ostáva otázkou, či existujú aspoň polynomiálne algoritmy na zistenie cyklu zápornej dĺžky vo všeobecne ohodnotenom digrafe. Odpoveď na túto otázku dáva nasledujúci algoritmus:

Algoritmus 3.4. Algoritmus na hľadanie cyklu zápornej dĺžky vo všeobecne ohodnotenom digrafe.

Krok 1: Inicializácia. Polož $t(i) := 0$, $x(i) := 0$ pre všetky $i \in V$.
Polož $\mathcal{E} := V$.

Krok 2: Vyber $r \in \mathcal{E}$, polož $\mathcal{E} := \mathcal{E} - \{r\}$.
Pre všetky hrany tvaru $(r, j) \in H$ urob:
Ak $t(j) > t(r) + c(r, j)$, potom

- $t(j) := t(r) + c(r, j)$, $x(j) := r$, $\mathcal{E} := \mathcal{E} \cup \{j\}$.
- polož $k := 1$, $p_1 := j$
  do $p_{k+1} := x(p_k)$, $k := k + 1$
  while ($x(p_k) \neq 0$ AND $x(p_k) \neq j$)
  - Ak $p_k = 0$, preznačením nevznikol cyklus. Pokračuj krokom 3.
  - Ak $p_k = j$, potom sme našli cyklus zápornej dĺžky. Je to cyklus určený vrcholmi $j = p_k, p_{k-1}, \ldots, p_1 = j$. STOP.

Krok 3: Ak $\mathcal{E} \neq \emptyset$, choď na Krok 2.
Ak $\mathcal{E} = \emptyset$, potom v digrafe G neexistuje cyklus zápornej dĺžky. STOP.

3.2.3 Výpočet matice vzdialeností

Kilometrovník – matica vzdialeností je často potrebným údajom v dopravnej i skladovej logistike. Možno ju vypočítať opakovaním Dijkstrovho algoritmu (alebo iného podobného algoritmu) postupne pre všetky $u \in V$. Tento postup bol v pionierskych časoch výpočtovej techniky preferovaný, lebo šetril pamäť i výpočtový čas.

Nasledujúci algoritmus počíta naraz celú maticu vzdialeností. Pre svoju jednoduchosť je mimoriadne ľahko implementovateľný.

Algoritmus 3.5. Floydov algoritmus na výpočet matice vzdialeností vrcholov v hranovo ohodnotenom grafe alebo digrafe $G = (V, H, c)$, kde $c(h) \geq 0$.

Krok 1: Zostroj maticu $\mathbf{C} = (c_{ij})$, ktorej prvky sú definované takto:

$$c_{ii} = 0 \quad \text{pre všetky} \quad i \in V$$

a pre všetky $i,\ j$, také, že $i \neq j$

$$c_{ij} = \begin{cases} c(i,j), & \text{ak } \{i,j\} \in H, \text{ resp. } (i,j) \in H \\ \infty, & \text{ak } \{i,j\} \notin H, \text{ resp. } (i,j) \notin H \end{cases}$$

a maticu $\mathbf{X}$

$$x_{ii} = i \quad \text{pre všetky} \quad i \in V$$

a pre všetky $i,\ j$, také, že $i \neq j$

$$x_{ij} = \begin{cases} i \quad , & \text{ak } \{i,j\} \in H, \text{ resp. } (i,j) \in H \\ \infty, & \text{ak } \{i,j\} \notin H, \text{ resp. } (i,j) \notin H \end{cases}$$

Krok 2: Urob pre všetky $k = 1, 2, \ldots, n = |V|$:
Pre všetky $i \neq k$ také, že $c_{ik} \neq \infty$, a pre všetky $j \neq k$ také, že $c_{kj} \neq \infty$, urob:
Ak $c_{ij} > c_{ik} + c_{kj}$, potom polož:

$$c_{ij} := c_{ik} + c_{kj}$$
$$x_{ij} := x_{kj}$$

Po skončení Floydovho algoritmu je matica $\mathbf{C}$ maticou vzdialeností vrcholov grafu, resp. digrafu G a matica $\mathbf{X}$ obsahuje pre každú dvojicu vrcholov i, j predposledný vrchol najkratšej i–j cesty. Ak potrebujeme nájsť najkratšiu i–j cestu, využijeme maticu smerníkov $\mathbf{X}$ takto: Pre predposledný vrchol j_1 najkratšej i–j cesty je $j_1 = x_{ij}$. Ďalší vrchol tejto cesty (odzadu) je $j_2 = x_{ij_1}$, ďalší $j_3 = x_{ij_2}$ atď., pokiaľ nedôjdeme do vrchola i.

V prípade, že potrebujeme len vzdialenosti a nepotrebujeme rekonštruovať príslušné najkratšie cesty, možno algoritmus použiť bez počítania matice $\mathbf{X}$.

3.2.4 Najspoľahlivejšia cesta v dopravnej sieti

Majme úseky dopravnej siete ohodnotené ich spoľahlivosťou. *Spoľahlivosť úseku* je definovaná ako pravdepodobnosť jeho úspešného prechodu. Modelom príslušnej siete je digraf alebo graf $G = (V, H, c)$ kde ohodnotenie $c(h)$ predstavuje spoľahlivosť hrany h (pravdepodobnosť úspešného prechodu hranou), t. j. $0 \leq c(h) \leq 1$. Nech $m(u, v)$ je u–v cesta. *Spoľahlivosť cesty* $m(u, v)$ – $s(m(u, v))$ – definujeme nasledujúco:

$$s(m(u,v)) = \prod_{h \in m(u,v)} c(h)$$

Hľadáme cestu, ktorá má maximálnu spoľahlivosť $s(m(u, v))$.

Pretože je funkcia $\log(x)$ (desiatkový logaritmus) rastúcou funkciou, je spoľahlivosť cesty $m(u, v)$ $s(m(u, v)) = \prod_{h \in m(u,v)} c(h)$ maximálna práve vtedy, keď je maximálny výraz:

$$\log\left(s(m(u,v)\right) = \log\left(\prod_{h \in m(u,v)} c(h)\right) = \sum_{h \in m(u,v)} \log\left(c(h)\right)$$

Posledný výraz je maximálny práve vtedy, keď je súčet $\sum_{h \in m(u,v)} - \log\left(c(h)\right)$ minimálny. Pretože $0 < c(h) \leq 1$, je $-\log\left(c(h)\right)$ nezáporné číslo. Pre cestu $m(u, v)$ je $\sum_{h \in m(u,v)} - \log\left(c(h)\right)$ vlastne dĺžkou cesty v hranovo ohodnotenom grafe $G = (V, H, c)$ s cenou hrany $\overline{c}(h) = -\log\left(c(h)\right)$.

Z posledných úvah vyplýva nasledujúci postup: u–v cestu maximálnej spoľahlivosti v grafe, resp. digrafe $G = (V, H, c)$ hľadáme ako najkratšiu cestu v grafe, resp. digrafe $\overline{G} = (V, H, \overline{c})$, kde pre cenu hrany $\overline{c}$ platí $\overline{c}(i, j) = -\log\left(c(i, j)\right)$.

Pre popísanú metódu môžeme použiť namiesto desiatkového logaritmu logaritmus pri akomkoľvek základe $z > 1$.

3.2.5 Cesta maximálnej priepustnosti

Pri preprave nadrozmerných nákladov je nutné navrhnúť cestu od odosielateľa k príjemcovi, ktorá obsahuje úseky s dostatočnou priepustnosťou (šírkou cesty, nosnosťou mostov, výškou podjazdov, polomerom zatáčok atď.). *Priepustnosť* takejto cesty je daná minimom priepustnosti jej úsekov. Stojíme teraz pred problémom nájsť cestu s maximálnou priepustnosťou. Metódy teórie grafov pomáhajú riešiť i tieto logistické problémy.

Matematickým modelom dopravnej siete s kapacitne obmedzenými úsekmi je hranovo ohodnotený graf $G = (V, H, c)$, v ktorom $c(h) > 0$ znamená *priepustnosť hrany* h. *Priepustnosť* $c(m(u, v))$ *u–v cesty (sledu, polosledu, atď.)* $m(u, v)$ definujeme ako

$$c(m(u, v)) = \min_{h \in m(u,v)} \{c(h)\},$$

t. j. ako minimum z priepustností všetkých hrán cesty (sledu, polosledu atď.)

Na hľadanie cesty maximálnej priepustnosti budeme potrebovať niekoľko nasledujúcich pojmov:
Strom je súvislý graf, ktorý neobsahuje cyklus.
Kostra súvislého grafu $G = (V, H)$ je taký jeho podgraf obsahujúci všetky vrcholy z V, ktorý je stromom.
Nech $G = (V, H, c)$ je hranovo ohodnotený graf, K kostra grafu G. *Cena* $c(K)$ *kostry* K je súčet ohodnotení jej hrán.
Najlacnejšia kostra v grafe G je kostra s najmenšou cenou.
Najdrahšia kostra v grafe G je kostra s najväčšou cenou.

Algoritmus 3.6. Kruskalov algoritmus na hľadanie najdrahšej kostry súvislého hranovo ohodnoteného grafu $G = (V, H, c)$.

Krok 1: Zoraď hrany podľa ich ohodnotenia zostupne do postupnosti $\mathcal{P}$.

Krok 2: Nech prvá hrana v postupnosti $\mathcal{P}$ je hrana $\{u, v\}$.
Vylúč hranu $\{u, v\}$ z postupnosti $\mathcal{P}$ a ak s už vybranými hranami nevytvára cyklus, zaraď ju do kostry.

Krok 3: Ak je počet vybraných hrán rovný $|V|-1$, STOP. Inak GOTO Krok 2.

Ak v kroku 1 zoradíme hrany grafu podľa ich ohodnotenia vzostupne, výsledkom algoritmu bude najlacnešia kostra grafu G.

Dá sa ukázať, že ak K je najdrahšia kostra v súvislom hranovo ohodnotenom grafe $G = (V, H, c)$, potom pre ľubovoľné dva vrcholy $u, v \in V$ je (jediná) u–v cesta v K u–v cestou maximálnej priepustnosti v G. S využitím tejto skutočnosti možno navrhnúť nasledujúci algoritmus:

Algoritmus 3.7. Algoritmus na hľadanie u–v cesty maximálnej priepustnosti v súvislom hranovo ohodnotenom grafe $G = (V, H, c)$.

Krok 1: V grafe G zostroj najdrahšiu kostru K.

Krok 2: Pre ľubovoľné dva vrcholy $u, v \in V$ je (jediná) u–v cesta v K u–v cestou maximálnej priepustnosti v G.

Uvedený algoritmus síce nájde u–v cestu maximálnej priepustnosti, no táto nemusí byť a spravidla ani nebýva optimálnou z hľadiska prejdenej vzdialenosti. Ak by sme chceli nájsť najkratšiu u–v cestu s maximálnou priepustnosťou, potrebujeme mať v príslušnom grafe okrem kapacitného ohodnotenia hrán aj ohodnotenie vyjadrujúce ich dĺžku.

Algoritmus 3.8. Algoritmus na hľadanie najkratšej u–v cesty s maximálnou priepustnosťou v súvislom hranovo ohodnotenom grafe $G = (V, H, c, d)$, kde $c(h)$ je priepustnosť a $d(h)$ je dĺžka hrany $h \in H$.

Krok 1: V grafe G nájdi cestu $m(u, v)$ maximálnej priepustnosti vzhľadom na ohodnotenie hrán c. Nech C je priepustnosť cesty $m(u, v)$.

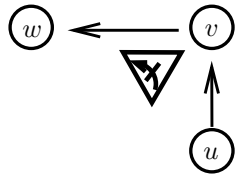
Krok 2: Vytvor graf $G' = (V, H', d)$, kde $H' = \{h | h \in H,\ c(h) \geq C\}$ (H' obsahuje len tie hrany pôvodného grafu, ktoré majú priepustnosť väčšiu alebo rovnú než C).

Krok 3: V grafe G' nájdi najkratšiu u-v cestu vzhľadom na ohodnotenie hrán d.

3.2.6 Modelovanie zakázaných odbočení v dopravnej sieti

Pri riešení praktických optimalizačných úloh v dopravných sieťach (napr. Janáček [23], Jánošíková [24]) sa môžeme stretnúť s potrebou zohľadniť bežné obmedzenie pohybu dopravných prostriedkov – zakázané odbočovanie. Takýchto zakázaných odbočení (najmä ľavých) v cestnej doprave stále pribúda s jej rastúcou intenzitou, v koľajovej doprave je bežné a vyplýva z jej technológie. Dostupné počítačové systémy na výpočet optimálnych trás (napr. AutoRoute) ani špecializované programy (napr. [14], Jánošíková a kol. [21]) však s týmto obmedzením nepočítajú.

Problém najkratšej trasy z miesta u do miesta v v dopravnej sieti sme modelovali v časti 3.2.1 (str. 82) ako úlohu o najkratšej u–v ceste v hranovo ohodnotenom digrafe a na jej riešenie sme uviedli niekoľko algoritmov.

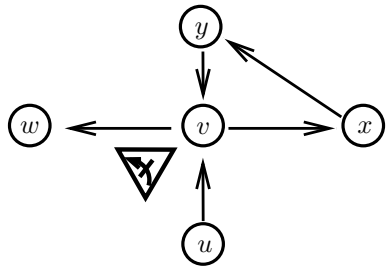


Obr. 3.2: Zakázané ľavé odbočenie

Zákaz odbočenia v niektorom uzle v dopravnej siete môžeme slovne popísať takto: Ak prichádzaš do križovatky v siete smerom od križovatky u, nesmieš z križovatky v pokračovať do križovatky w – obrázok 3.2. V reči teórie grafov: Cesta (resp. ťah alebo sled) rešpektujúci zakázané odbočenie nesmie obsahovať bezprostredne za sebou hrany (u, v) a (v, w) – ešte presnejšie, sled ako striedavá postupnosť vrcholov a hrán nesmie obsahovať podpostupnosť $(u, v), v, (v, w)$.

Zakázané odbočenia môžeme teda popísať množinou zakázaných usporiadaných dvojíc susedných hrán. Sled (resp. ťah, cestu) rešpektujúci zakázané odbočenia nazveme prípustným sledom v dopravnej sieti so zakázanými odbočeniami.

Poznámka k notácii. Pre rozlíšenie usporiadanej dvojice hrán od samotnej hrany digrafu G (ktorá je definovaná ako usporiadaná dvojica vrcholov grafu G) budeme usporiadanú dvojicu hrán vkladať do hranatých zátvoriek – $[(u, v), (v, w)]$.



Obr. 3.3: Digraf so zakázaným odbočením
V tomto digrafe neexistuje prípustná u-w cesta.
Prípustný u–w sled je určený vrcholmi $u,\ v,\ x,\ y,\ v,\ w$.

Hľadanie najkratšej prípustnej cesty v digrafe so zakázanými odbočeniami je už úloha zložitejšia ako hľadanie najkratšej cesty. Ako vidieť z obr. 3.3 prípustná u–w cesta neexistuje. Musíme teda hľadať najkratší prípustný u–w sled.

Úlohu o najkratšej trase so zakázaným odbočovaním vozidiel môžeme formulovať takto:
Nech je daný hranovo ohodnotený digraf $G = (V, H, c)$ s množinou vrcholov V, s množinou orientovaných hrán H a ohodnotením hrán c predstavujúcim ich dĺžku.
Nech je daná množina zakázaných odbočení tvorená niektorými usporiadanými dvojicami susedných hrán

$$\mathcal{Z} = \big\{[(u,v),(v,w)] \mid (u,v) \in H,\ (v,w) \in H,\ [(u,v),(v,w)] \text{ je zakázané odbočenie}\big\}\ .$$

Potom prípustným u–v sledom v digrafe G so zakázaným odbočovaním $\mathcal{Z}$ rozumieme striedavú postupnosť vrcholov a hrán začínajúcu vo vrchole u a končiacu vo vrchole v.

$$u = v_1, (v_1, v_2), v_2, (v_2, v_3), v_3, \ldots, (v_{n-1}, v_n), v_n = v,$$

kde $(v_{i-1}, v_i) \in H$ pre $i = 2, 3, \ldots, n$ a kde $[(v_{i-1}, v_i), (v_i, v_{i+1})] \notin \mathcal{Z}$ pre $i = 2, 3, \ldots, n-1$.
Našou úlohou je teda nájsť pre dané koncové vrcholy u a v najkratší prípustný u–v sled v digrafe so zakázaným odbočovaním $\mathcal{Z}$.

Označme symbolom I_V množinu všetkých usporiadaných dvojíc typu (v, v), kde $v \in V$, t. j.

$$I_V = \{(v, v) \mid v \in V\}.$$

Zostrojme hranovo ohodnotený digraf $G_H = (V_H, E_H, d)$ s množinou vrcholov V_H:

$$V_H = H \cup I_V \tag{3.1}$$

a množinou orientovaných hrán E_H:

$$\begin{aligned} E_H = {} & \{[(x,y),(y,z)] \mid (x,y) \in H,\ (y,z) \in H\} \cup \\ & \cup \{[(y,y),(y,z)] \mid (y,z) \in H\} \cup \{[(x,y),(y,y)] \mid (x,y) \in H\} - \mathcal{Z}. \end{aligned} \tag{3.2}$$

Množina vrcholov V_H je tvorená všetkými hranami pôvodného digrafu a naviac všetkými usporiadanými dvojicami typu (v, v). Množina orientovaných hrán E_H je tvorená množinou usporiadaných dvojíc $[(u, v), (v, w)]$ susedných hrán

pôvodného digrafu G, ktoré nie sú zakázanými odbočovaniami $\mathcal{Z}$ a množinou všetkých usporiadaných dvojíc typu $[(x,x),(xy)]$ a typu $[(x,y),(y,y)]$, kde $x \in V$, $y \in V$, $(x,y) \in H$. Každá hrana z E_H je teda usporiadanou dvojicou typu $[(x,y),(y,z)]$.
Ohodnotenie orientovaných hrán $d : E \to R$ definujeme pomocou ohodnotenia c orientovaných hrán digrafu G takto:

$$d[(x,y),(y,z)] = \begin{cases} 0 & \text{ak } x = y \\ c(x,y) & \text{ak } x \neq y \end{cases} \tag{3.3}$$

Majme (u,u)–(v,v) cestu v digrafe G_H. Táto cesta má podľa definície tvar:

$$(u,u) = h_1, [h_1,h_2], h_2, ..., [h_{n-1},h_n], h_n = (v,v), \tag{3.4}$$

kde $[h_{i-1},h_i] \in E_H$ pre $i = 2,3,...,n-1$ a v ktorej sa vrcholy neopakujú.
Dĺžka (u,u)–(v,v) cesty 3.4 je

$$d[h_1,h_2] + d[h_2,h_3] + \cdots + d[h_{n-1},h_n] \;=\; c(h_1) + c(h_2) + \cdots + c(h_{n-1}).$$

Majme prípustný u–v sled v digrafe G so zakázanými odbočeniami $\mathcal{Z}$:

$$v_1, (v_1,v_2), v_2, (v_2,v_3), v_3, \ldots, (v_{n-1},v_n), v_n, \tag{3.5}$$

kde $v_1 = u$, $v_n = v$.
Potom tomuto sledu zodpovedá sled v digrafe G_H:

$$\begin{aligned}(v_1,v_1), [(v_1,v_1),(v_1,v_2)], (v_1,v_2), [(v_1,v_2),(v_2,v_3)], (v_2,v_3), \ldots, \\ [(v_{n-2},v_{n-1}),(v_{n-1},v_n)], (v_{n-1},v_n), [(v_{n-1},v_n),(v_n,v_n)], (v_n,v_n).\end{aligned} \tag{3.6}$$

Oba sledy (3.5), (3.6) majú rovnakú dĺžku, a to $\sum_{i=1}^{n-1} c(v_i,v_{i+1})$.
Naopak, každému sledu (3.6) v digrafe G_H zodpovedá prípustný sled (3.5) v digrafe G so zakázanými odbočeniami $\mathcal{Z}$ s rovnakou dĺžkou.

Nájsť najkratší prípustný u–v sled v digrafe G so zakázanými odbočeniami $\mathcal{Z}$ znamená nájsť najkratšiu cestu v príslušnom digrafe G_H skonštruovanom podľa vzťahov (3.1), (3.2), (3.3). Úlohu hľadania najkratšieho prípustného sledu v digrafe so zakázanými odbočeniami sme tak previedli na klasickú úlohu hľadania najkratšej cesty, na ktorú môžeme použiť príslušné známe algoritmy.

Dá sa ukázať, že výsledný optimálny sled v pôvodnom digrafe G je ťahom – neobsahuje žiadnu hranu dvakrát.

Na tomto mieste je nutné spomenúť, že uvedený postup nie je prvým pokusom o modelovanie trás so zakázaným odbočovaním. Je známy model pomocou tzv. polárnych grafov (napr. Zelinka [55]). Model s polárnymi grafmi však naviac vyžaduje tvorbu špeciálneho grafového algoritmu a pri praktickej realizácii aj následnú tvorbu zvláštneho programu.
Iný prístup pomocou multi–polarizovaných grafov uvádza Černý v [6]. Náš postup previedol daný problém na klasickú úlohu o najkratšej ceste bez nutnosti zavádzania nových pojmov a konštrukcie špeciálneho algoritmu.

Navrhnutý postup bol úspešne použitý pri modelovaní sietí autobusovej dopravy niekoľkých miest a tiež na električovú dopravu v Prahe. Oceňovanou vlastnosťou tvorby optimálnych trás bolo modelovanie otáčania sa električiek na konečných zastávkach liniek. Prinieslo totiž nečakane vysokú 24% priemernú úsporu vzdialenosti vzhľadom na rutinné riešenia.

3.3 Toky v sieťach

V tejto časti budeme pod *sieťou* rozumieť hranovo ohodnotený digraf $G = (V, H, c, d)$, v ktorom ohodnotenie $c(h) > 0$ každej orientovanej hrany $h \in H$ je celočíselné a predstavuje priepustnosť hrany h, $d(h) > 0$ je dĺžka orientovanej hrany, a v ktorom existuje práve jeden vrchol z taký, z ktorého hrany len vychádzajú a práve jeden vrchol u taký, že do ktorého hrany len vchádzajú. Vrchol z nazveme *zdroj*, vrchol u nazveme *ústie*.

Tokom v sieti $G = (V, H, c)$ nazveme celočíselnú funkciu $\mathbf{y} : H \to \mathbb{R}$ definovanú na množine orientovaných hrán H, pre ktorú platí:

$$1.\quad \mathbf{y}(i,j) \geq 0 \quad \forall (i,j) \in H \tag{3.7}$$

$$2.\quad \mathbf{y}(i,j) \leq c(i,j) \quad \forall (i,j) \in H \tag{3.8}$$

$$3.\quad \sum_{\substack{i \\ (i,v)\in H}} \mathbf{y}(i,v) = \sum_{\substack{j \\ (v,j)\in H}} \mathbf{y}(v,j) \quad \forall v \in V,\ v \neq u,\ v \neq z \tag{3.9}$$

$$4.\quad \sum_{\substack{j \\ (z,j)\in H}} \mathbf{y}(z,j) = \sum_{\substack{i \\ (i,u)\in H}} \mathbf{y}(i,u) \tag{3.10}$$

Veľkosťou $F(\mathbf{y})$ toku $\mathbf{y}$ nazveme číslo (3.10) Hovoríme, že tok $\mathbf{y}$ v sieti G je *maximálny*, ak má najväčšiu veľkosť zo všetkých možných tokov v sieti G. *Cenou*

toku **y** nazveme číslo

$$D(\mathbf{y}) = \sum_{h \in H} d(h).\mathbf{y}(h) = \sum_{\substack{i,j \\ (i,j) \in H}} \mathbf{d}(i,j).y(i,j)$$

Majme sieť $G = (V, H, c, d)$ s tokom **y**, v, $w \in V$. Nech $m(v,w)$ je v–w polocesta v G, nech h je orientovaná hrana tejto polocesty. *Rezervu hrany* v poloceste $m(v,w)$ definujeme nasledujúco:

$$r(h) = \begin{cases} c(h) - \mathbf{y}(h) & \text{ak je hrana } h \text{ použitá v } m(v,w) \text{ v smere orientácie} \\ \mathbf{y}(h) & \text{ak je hrana } h \text{ použitá v } m(v,w) \text{ proti smeru orientácie} \end{cases} \tag{3.11}$$

Rezerva polocesty (resp. *rezerva polocyklu*) $m(v,w)$ je minimum rezerv hrán tejto polocesty (resp. polocyklu). Hovoríme, že polocesta zo zdroja do ústia $m(z,u)$ je *zlepšujúca polocesta*, ak má kladnú rezervu.
Hovoríme, že polocyklus $\mathcal{C}$ je *zlepšujúci polocyklus* , ak má kladnú rezervu a ak

$$L(\mathcal{C}) = \sum_{\substack{h,\ h \in \mathcal{C} \\ h \text{ je v smere} \\ \text{orientácie}}} d(h) \ - \sum_{\substack{h,\ h \in \mathcal{C} \\ h \text{ je proti} \\ \text{smeru orientácie}}} d(h) < 0$$

Platí nasledujúca Fordova Fulkersonova veta:

Veta 3.1. *Tok* **y** *v sieti* G *je maximálny práve vtedy, keď v sieti* G *s tokom* **y** *neexistuje zlepšujúca polocesta.*
Tok **y** *je tokom s minimálnou cenou medzi všetkými tokmi veľkosti* $F(\mathbf{y})$ *práve vtedy, keď v sieti* G *s tokom* **y** *neexistuje zlepšujúci polocyklus.*

Algoritmus 3.9. Ford – Fulkersonov algoritmus na hľadanie maximálneho toku v sieti G=(V,H,c).

Krok 1: Zvoľ v sieti počiatočný tok **y**, napríklad nulový tok.

Krok 2: Nájdi v sieti G s tokom **y** zväčšujúcu polocestu $m(z,u)$.

Krok 3: Ak zväčšujúca polocesta neexistuje, tok **y** je maximálny. STOP.

Krok 4: Ak zväčšujúca polocesta $m(z,u)$ existuje zmeň tok **y** nasledujúco:

$$\mathbf{y}(h) := \begin{cases} \mathbf{y}(h) & \text{ak } h \text{ neleží na ceste } m(z,u) \\ \mathbf{y}(h) + r & \text{ak } h \text{ leží na ceste } m(z,u) \text{ v smere svojej orientácie} \\ \mathbf{y}(h) - r & \text{ak } h \text{ leží na ceste } m(z,u) \text{ proti smeru svojej orientácie} \end{cases}$$

GOTO Krok 2.

Algoritmus 3.10. Algoritmus na hľadanie najlacnejšieho toku danej veľkosti v sieti $G = (V, H, c, d)$**.**

Krok 1: Začni tokom $\mathbf{y}$ v sieti $G = (V, H, c, d)$ danej veľkosti.

Krok 2: V sieti G s tokom $\mathbf{y}$ nájdi zlepšujúci polocyklus C.

Krok 3: Ak zlepšujúci polocyklus neexistuje, tok $\mathbf{y}$ je najlacnejší zo všetkých tokov svojej veľkosti. STOP.

Krok 4: Ak $\mathcal{C}$ je zlepšujúci polocyklus, zmeň tok $\mathbf{y}$ nasledujúco:

$$\mathbf{y}(h) := \begin{cases} \mathbf{y}(h) & \text{ak } h \text{ neleží na polocykle } \mathcal{C} \\ \mathbf{y}(h) + r & \text{ak } h \text{ leží na polocykle } \mathcal{C} \text{ v smere svojej orientácie} \\ \mathbf{y}(h) - r & \text{ak } h \text{ leží na polocykle } \mathcal{C} \text{ proti smeru svojej orientácie} \end{cases}$$

GOTO Krok 2.

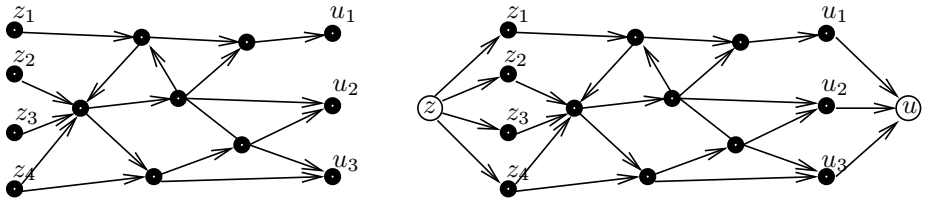
3.4 Siete s viacerými zdrojmi a ústiami

V praktickom živote sa stretávame aj so sieťami, ktoré majú mnoho zdrojov a mnoho ústí. Takúto situáciu možno modelovať sieťou G s niekoľkými zdrojmi a ústiami. Aby sme túto situáciu prispôsobili nášmu modelu s jedným zdrojom a jedným ústím, pridáme ku grafu G jeden fiktívny zdroj a jedno fiktívne ústie. Od zdroja vedieme orientované hrany k všetkým zdrojom siete G. Kapacitu hrany typu (fiktívny zdroj, zdroj) môžeme definovať ako nekonečnú, ale ponúka sa aj možnosť definovaním konečnej kapacity tejto hrany definovať kapacitu príslušného zdroja. Podobne vedieme orientované hrany s neohraničenou kapacitou (alebo kapacitou príslušného ústia) od všetkých ústí siete G k fiktívnemu ústiu. Dostávame tak sieť v našom doterajšom zmysle, v ktorej môžeme vyriešiť všetky kapacitné a optimalizačné problémy týkajúce sa siete G.

3.5 Dopravná úloha a jej varianty

3.5.1 Formulácia klasickej dopravnej úlohy

Máme p dodávateľov $D_1, D_2, \ldots, D_p$ s kapacitami $a_1, a_2, \ldots, a_p$ a q odberateľov $O_1, O_2, \ldots, O_q$ s požiadavkami $b_1, b_2, \ldots, b_q$. Náklady na prepravu jednotky tovaru od dodávateľa D_i k odberateľovi O_j sú d_{ij}. Našou úlohou je určiť, koľko



Obr. 3.4: Pridaním fiktívneho zdroja a fiktívneho ústia
k sieti s viacerými zdrojmi a ústiami
dostaneme sieť s jedným zdrojom a jedným ústím

tovaru od ktorého dodávateľa ku ktorému odberateľovi doviezť tak, aby sme rozviezli čo najviac tovaru tak, aby ani kapacity dodávateľov, ani požiadavky odberateľov neboli prekročené a tak, aby celková cena za prepravu všetkého tovaru bola čo najmenšia. Takto formulovaná úloha sa volá *dopravná úloha*.

3.5.2 Model lineárneho programovania pre dopravnú úlohu

Pre dopravnú úlohu existuje niekoľko matematických modelov. Model lineárneho programovania je nasledujúci.

Predpokladajme najprv, že súčet požiadaviek odberateľov je rovný súčtu kapacít dodávateľov, t. j. $\sum_{i=1}^{p} a_i = \sum_{j=1}^{q} b_j$. Takáto dopravná úloha sa nazýva *vybilancovaná*. Nech $\mathbf{X}$ je matica reálnych čísel typu $p \times q$, ktorej prvok x_{ij} znamená množstvo tovaru dovezené od dodávateľa D_i k odberateľovi O_j. Potom celková cena za prepravu všetkého tovaru bude $\sum_{i=1}^{p} \sum_{j=1}^{q} d_{ij} x_{ij}$.

Vyriešiť dopravnú úlohu znamená určiť prvky matice $\mathbf{X}$ tak, aby

$$\sum_{i=1}^{p} \sum_{j=1}^{q} d_{ij} x_{ij} \text{ bolo minimálne} \tag{3.12}$$

za predpokladov

$$\sum_{j=1}^{q} x_{ij} = a_i \quad \forall i = 1, 2, \ldots, p \tag{3.13}$$

$$\sum_{i=1}^{p} x_{ij} = b_j \quad \forall j = 1, 2, \ldots, q \tag{3.14}$$

$$x_{ij} \geq 0 \quad \forall i = 1, 2, \ldots, p \quad \forall j = 1, 2, \ldots, q \tag{3.15}$$

Dvojitá suma v (3.12) znamená celkové prepravné náklady, podmienka (3.13) znamená, že od každého dodávateľa D_i prepravíme celé ponúkané množstvo a_i, podmienka (3.14) znamená, že každému odberateľovi dovezieme všetko požadované množstvo tovaru b_j. Podmienka (3.15) hovorí, že nemožno prepravovať záporné množstvá tovaru.

Ak by ponuka dodávateľov bola väčšia než požiadavky odberateľov, t. j. $\sum_{i=1}^{p} a_i > \sum_{j=1}^{q} b_j$, model úlohy sa zmení tak, že v podmienkach (3.13) bude namiesto „=“ vzťah „≤“, t. j. od každého dodávateľa vyvezieme najviac a_i jednotiek tovaru. Ostatné obmedzenia (3.14), (3.15) ostávajú nezmenené. Podobne, ak $\sum_{i=1}^{p} a_i < \sum_{j=1}^{q} b_j$, potom sa v podmienkach (3.14) zmení „=“ na „≤“.

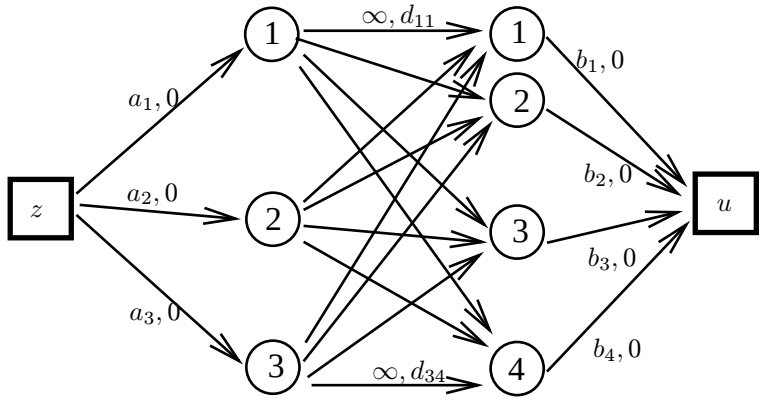
Niektoré metódy riešenia dopravnej úlohy predpokladajú jej vybilancovaný tvar. Nevybilancovanú úlohu prevedieme na vybilancovaný prípad tak, že ak súčet kapacít dodávateľov prevyšuje súčet požiadaviek odberateľov, t. j. ak $\sum_{i=1}^{p} a_i > \sum_{j=1}^{q} b_j$, potom dodáme fiktívneho odberateľa O_{q+1} s požiadavkou $b_{q+1} = \sum_{i=1}^{p} a_i - \sum_{j=1}^{q} b_j$ a všetkými prepravnými nákladmi $d_{i(q+1)}$ nulovými. Analogicky v prípade $\sum_{i=1}^{p} a_i < \sum_{j=1}^{q} b_j$ dodáme fiktívneho dodávateľa.

3.5.3 Model teórie grafov pre dopravnú úlohu

Na riešenie tejto úlohy zostrojme sieť G s množinou vrcholov $V = V_1 \cup V_2 \cup \{z, u\}$, kde V_1 je množina všetkých dodávateľov, V_2 množina všetkých odberateľov, z je fiktívny zdroj a u fiktívne ústie. Množinu hrán H siete G budú tvoriť všetky hrany typu (z, D_i) s kapacitou a_i a nulovou cenou za prepravu jednotky tovaru, všetky hrany typu (O_j, u) s kapacitou b_j a nulovou cenou za prepravu jednotky tovaru a nakoniec všetky hrany typu (D_i, O_j) s neobmedzenou kapacitou a cenou za prepravu jednotky tovaru rovnou d_{ij}. Príklad grafu G pre 3 dodávateľov a 4 odberateľov je na obrázku 3.5.

Ak máme v sieti G tok $\mathbf{y}$, potom veľkosť toku $\mathbf{y}$ je celkové množstvo tovaru prepravené od všetkých dodávateľov ku všetkým odberateľom a cena toku $\mathbf{y}$ je rovná príslušným dopravným nákladom Ak v takto definovanej sieti G nájdeme najlacnejší maximálny tok $\mathbf{y}$, potom

- požiadavka maximality toku $\mathbf{y}$ zaistí, že sa prepraví $\min\left\{\sum_{i=1}^{p} a_i, \sum_{j=1}^{q} b_j\right\}$ jednotiek tovaru
- skutočnosť, že $\mathbf{y}$ je najlacnejší maximálny tok zaistí, že celkové prepravné náklady budú najnižšie



Obr. 3.5: Model teórie grafov pre dopravnú úlohu
s troma dodávateľmi a štyrmi odberateľmi

- Hodnoty $\mathbf{y}(D_i, O_j)$ pre všetky $i = 1, 2, \ldots, p$, $j = 1, 2, \ldots, q$ sú optimálne množstvá tovaru, ktoré treba doviezť od dodávateľa D_i k odberateľovi O_j.

3.5.4 Metódy riešenia dopravnej úlohy

Na riešenie dopravnej úlohy formulovanej vzťahmi (3.12) – (3.15) sa používa aparát lineárneho programovania. Z neho vyplýva, že matica optimálneho riešenia $\mathbf{X}$ dopravnej úlohy má najviac $p + q - 1$ nenulových prvkov. Teória lineárneho programovania ponúka na riešenie dopravnej úlohy modifikované verzie lineárneho programovania – napr. Dantzigova primárna metóda alebo maďarská duálna metóda – obe využívajúce špeciálny tvar príslušnej úlohy lineárneho programovania.

Autori však odporúčajú použiť na praktické výpočty algoritmy na všeobecné lineárne programovanie, ktoré sú bežne dostupné v komerčných i voľne použiteľných softvérových balíkoch a tiež v tabuľkových procesoroch a mnohé z nich zvládnu aj veľké praktické úlohy.

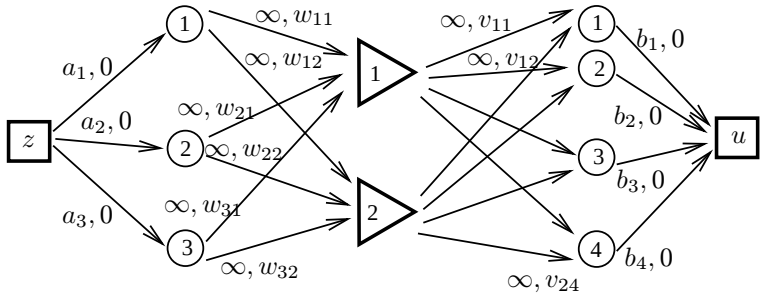
Pre grafovú formuláciu maximálny tok $\mathbf{y}$ nájdeme algoritmom 3.9 a príslušný najlacnejší tok algoritmom 3.10. Pre úlohy veľmi veľkého rozsahu máme najlepšie skúsenosti s formuláciou úlohy ako úlohy hľadania najlacnejšieho maximálneho toku v sieti. Vhodnou implementáciou tohto prístupu možno zvládnuť dopravné úlohy s tisíckami dodávateľov a odberateľov. Naviac toková formulá-

cia dopravnej úlohy umožňuje aj zovšeobecnenia dopravnej úlohy na zložitejšiu topológiu siete s viacerými úrovňami medziskladov a ďalšími špeciálnymi podmienkami.

3.5.5 Dopravná úloha s medziskladmi

V niektorých prípadoch tovar od prvotných dodávateľov (napr. výrobcov) nejde priamo k odberateľom, ale dopravuje sa najprv do medziskladov, odkiaľ sa expeduje k samotným odberateľom.

Máme p dodávateľov $D_1, D_2, \ldots, D_p$ s kapacitami $a_1, a_2, \ldots, a_p$ a q odberateľov $O_1, O_2, \ldots, O_q$ s požiadavkami $b_1, b_2, \ldots, b_q$. Prepravovaný tovar s musí najprv doviezť do r medziskladov $S_1, S_2, \ldots, S_r$. Náklady na prepravu jednotky tovaru od dodávateľa D_i k mezdiskladu S_j sú w_{ij}, do medziskladu S_j k odberateľovi O_k v_{jk}. Treba určiť množstvá tovaru x_{ij} prevezené od D_i k S_j a tiež množstvá yjk od S_j k O_k tak, aby sme rozviezli čo najviac tovaru tak, aby ani kapacity dodávateľov, ani požiadavky odberateľov neboli prekročené a tak, aby celková cena za prepravu všetkého tovaru bola čo najmenšia.



Obr. 3.6: Model teórie grafov pre dopravnú úlohu s dvoma medziskladmi s troma dodávateľmi a štyrmi odberateľmi

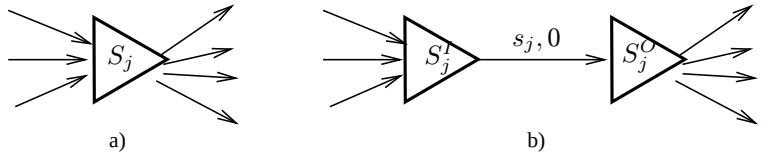
Ako model teórie grafov pre formulovanú úlohu bude digraf G s množinou vrcholov $V = V_1 \cup V_2 \cup V3 \cup \{z, u\}$, kde V_1 je množina všetkých dodávateľov, V_2 množina všetkých odberateľov, V_3 množina všetkých medziskladov, z je fiktívny zdroj a u fiktívne ústie. Množinu hrán H siete G budú tvoriť všetky hrany typu (z, D_i) s kapacitou a_i a nulovou cenou za prepravu jednotky tovaru, všetky hrany typu (O_j, u) s kapacitou b_j a nulovou cenou za prepravu jednotky tovaru a nakoniec všetky hrany typu (D_i, S_j) a (S_j, O_k) s neobmedzenou kapacitou

a cenou za prepravu jednotky tovaru rovnou w_{ij}, resp. v_{jk}. Príklad grafu G pre 3 dodávateľov a 4 odberateľov a 2 medziskladyje na obrázku 3.6.

Riešiť takto formulovanú úlohu znamená nájsť v sieti G najlacnejší maximálny tok.

Dodatočným výsledkom riešenia môže byť i požadovaná kapacita každého skladu S_j, ktorú dostaneme ako množstvo tovaru dovezeného do S_j od všetkých dodávateľov.

Práve popísaný model dovoľuje i modifikáciu, keď k formulácii úlohy dodáme ohraničenia na kapacity medziskladov. Nech medzisklad S_j má ohraničenú kapacitu s_j. Vtedy v grafárskom modeli G vrchol prislúchajúci medziskladu S_j nahradíme dvojicou vrcholov S_j^I – vstup skladu S_j, S_j^O – výstup skladu S_j, hrany typu (D_i, S_j) nahradíme hranami typu D_i, S_j^I, hrany typu (S_j, O_k) hranami S_j^O, O_k (oba typy s pôvodnou cenou a nekonečnou kapacitou) a naviac dodáme hranu (S_j^I, S_j^O) s nulovou cenou a kapacitou s_j.



Obr. 3.7: Modelovanie medziskladu
a) s neohraničenou kapacitou, b) s kapacitou s_j

Riešiť posledne formulovanú úlohu znovu znamená v sieti G nájsť maximálny tok s minimálnou cenou.

Teória grafov umožňuje modelovať najrozličnejšie varianty dopravnej úlohy s rôznou topológiou dopravnej siete, pričom všetky úlohy vedú k hľadaniu najlacnejšieho maximálneho toku v sieti.

3.6 Okružné dopravné problémy

Pod názvom *úlohy okružných jázd* - (VRP) Vehicle Routing Problems sa najčastejšie stretávame s rôznymi praktickými zovšeobecneniami *úlohy obchodného cestujúceho* – (TSP) Traveling Salesman Problem, ktorú môžeme formulovať takto: Je daná konečná množina miest a tabuľka vzdialeností medzi nimi. Cieľom obchodného cestujúceho je navštíviť postupne všetky mestá práve raz a vrátiť sa do východiskového mesta pri minimálnej prejdenej vzdialenosti.

Tieto úlohy vznikajú z praktických potrieb ďalších technologických obmedzení pri tvorbe trás vozidiel [22], ktoré obsluhujú niektoré vrcholy dopravnej siete. Vo firmách s vlastným vozidlovým parkom sa rozvoz a zvoz komodít (substrátu) realizuje neraz len na základe skúsenosti a intuície pomocou evidencie požiadaviek na prepravu a disponibilného počtu vozidiel. To spôsobuje neefektívne využitie vozidiel. V dôsledku neproduktívnych jázd potom vzniká aj potreba vyčleniť na požadovanú prácu väčší počet vozidiel než je nutné. Rozdiel skutočných nákladov od optimálnych býva až v rozmedzí do 30%.

Mnohé úspešné metódy riešenia úlohy VRP vychádzajú z princípov riešenia úlohy TSP [23]. Ich podrobnú klasifikáciu spolu s množstvom odkazov na literatúru možno nájsť v [1]. Ďalej sa obmedzíme hlavne na niektoré naše praktické skúsenosti s riešením týchto úloh.

Matematicky môžeme úlohu TSP formulovať takto: Je daná konečná množina $\mathcal{N} = \{1, 2, \ldots, n\}$ a reálna matica vzdialeností $\mathbf{C} \in \mathbb{R}^{n \times n}$. Hľadá sa taká cyklická permutácia π na množine $\mathcal{N}$, ktorá minimalizuje cieľovú funkciu $c(\pi)$

$$c(\pi) = \sum_{i \in \mathcal{N}} c_{i,\pi(i)}. \tag{3.16}$$

Cyklickej permutácii $\pi = \big(\pi(1), \pi(2), \ldots, \pi(n)\big)$ zodpovedá jediný cyklus C_π

$$C_\pi = 1 \to \pi(1) \to \pi(\pi(1)) \to \cdots \to \pi^n(1) = 1.$$

Požiadavka jediného cyklu v cyklickej permutácii π je podstatná. Ak by sme od nej upustili, dostali by sme riešenie známej polynomiálne riešiteľnej *priraďovacej úlohy* - (AP) Assignment Problem. Jednou z mnohých formulácií úlohy TSP je nasledujúca úloha bivalentného programovania (ATSP): Nájsť také x_{ij}, aby

$$\sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} c_{ij} x_{ij} \;\to\; \min \tag{3.17}$$

za podmienok:

$$\sum_{i \in \mathcal{N}} x_{ij} = 1 \qquad j \in \mathcal{N} \tag{3.18}$$

$$\sum_{j \in \mathcal{N}} x_{ij} = 1 \qquad i \in \mathcal{N} \tag{3.19}$$

$$\sum_{i \in S} \sum_{j \in \mathcal{N}-S} x_{ij} \geq 1 \qquad \forall S \subset \mathcal{N},\ 2 \leq |S| < |\mathcal{N}| \tag{3.20}$$

$$x_{ij} \in \{0, 1\} \qquad i, j \in \mathcal{N} \tag{3.21}$$

Tu je priraďovaciu úloha (3.17), (3.18), (3.19), (3.21) formulovaná pomocou rozhodovacích premenných x_{ij}. Podmienka (3.20) je jednou z možných *anticyklických podmienok*, ktoré zabezpečia, že vzniknutá permutácia $\pi, \pi(i) = j$ pre $x_{ij} = 1$ nebude obsahovať cykly dĺžky menšej než n.

Úloha TSP patrí medzi NP-ťažké kombinatorické úlohy, ktoré sú mimoriadne obťažné. V literatúre sa im venuje stála pozornosť, o čom svedčí aj vysoká citovanosť základnej monografie [30] z roku 1985. Doteraz nie je známy žiaden efektívny spôsob jej riešenia. Pri riešení pomocou dynamického programovania, ktoré požaduje $n.2^n$ pamäťových miest, boli exaktne riešené úlohy pre $n \leq 70$. Naša praktická skúsenosť s formuláciou úlohy ATSP (pre inštancie so symetrickými i nesymetrickými maticami vzdialeností) riešenej pomocou metódy vetiev a hraníc programovým balíkom *lp_solve* [2] nám umožnila vyriešiť úlohy pre $n \leq 50$, pričom sa nám osvedčilo postupné pridávanie anticyklických podmienok (3.20). Vyskytli sa však aj prípady, keď sa nám takýmto postupom nepodarilo nájsť optimálne riešenie.

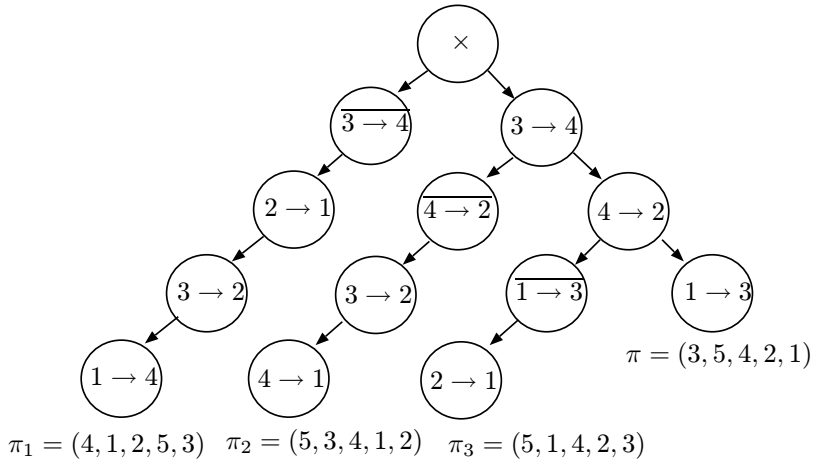
V prípade úloh väčších rozmerov ($n > 70$) nepredvídateľnosť polynomiálnej doby výpočtu spôsobuje, že *presné* metódy riešenia sú prakticky ťažko použiteľné a treba sa uspokojiť s *heuristickými* metódami, ktoré spravidla dávajú dostatočne presné *aproximatívne - suboptimálne* riešenie. Heuristické metódy sa delia na *tvoriace* (s cieľom nájsť nejaké „dobré“ prípustné riešenie) a *zlepšujúce* (s cieľom nájsť úpravou daného riešenia lacnejšie riešenie).

Pre TSP je asi najznámejšou tvoriacou heuristikou pomerne jednoduchá *metóda Clarka a Wrighta* (*greedy - pažravá*), našla uplatnenie aj vo VRP. Štartuje z neprípustného riešenia $n \to 1 \to n, n \to 2 \to n, \ldots, n-1 \to n \to n-1$, ktoré je tvorené množinou $n-1$ cyklami obsahujúcimi východiskové mesto TSP, v našom prípade n. V ďalších krokoch sa vždy nájde najvýhodnejšia dvojica cyklov $n \to \cdots \to i \to n$ a $n \to j \to \cdots \to n$, ktorá maximalizuje zisk $s_{ij} = c_{in} + c_{nj} - c_{ij}$, ak bude táto dvojica cyklov nahradená jediným cyklom $n \to \cdots \to i \to j \cdots \to n$. Po $n-1$ krokoch dostávame jediný cyklus, ktorý sa prehlási za suboptimálne riešenie metódy.

Existuje viacero zlepšení tejto „greedy“ metódy [50]. Osvedčila sa nám aplikácia **Holmesovej a Parkerovej perturbačnej schémy** [39], v ktorej je „greedy“ riešenie testované postupným dočasným zakazovaním jeho prejazdov medzi mestami.

Algoritmus 3.11. Holmesova a Parkerova perturbačná schéma

Krok 1: Nech $\mathcal{Z} = \emptyset$ je zoznam trvalo zakázaných jázd.



Obr. 3.8: Príklad perturbačnej schémy pre úlohu TSP s 5 mestami

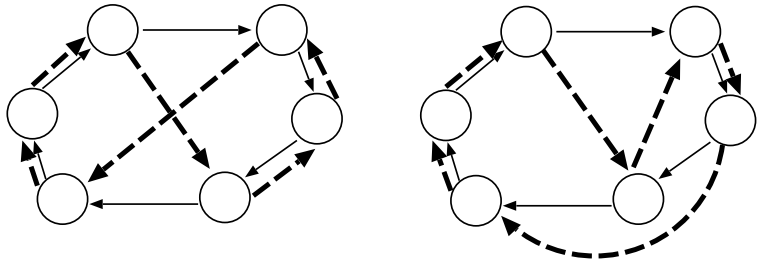
Krok 2: Vytvoríme cyklickú permutáciu π nejakou tvoriacou heuristikou tak, že prejazdy $\{i_1 \to \pi(i_1), i_2 \to \pi(i_2), \ldots, i_n \to \pi(i_n)\}$ neobsahujú zakázané jazdy z množiny $\mathcal{Z}$.

Krok 3: Pre $k = 1, 2, \ldots, n-2$ je zvolenou tvoriacou heuristikou vytvorená cyklická permutácia π_k, ktorá neobsahuje žiadne zakázané jazdy z množiny $\mathcal{Z} \cup \{i_k \to \pi(i_k)\}$.

Krok 4: Ak $c(\pi_*) = \min\{c(\pi_k) \mid k = 1, 2, \ldots, n-2\} < c(\pi)$, potom položíme $\pi = \pi_*$, $\mathcal{Z} = \mathcal{Z} \cup \{i_* \to \pi(i_*)\}$ a **GOTO Krok 2**. Ináč **STOP** s aproximatívnym riešením π.

Z ilustračného príkladu na obr. 3.8 vidíme, že pri hľadaní nových permutácií π_k v **Kroku 3** môžeme využiť časť testovaného riešenia π. Symbolom $\overline{u \to v}$ tu označujeme, že jazda $u \to v$ je dočasne zakázaná. Počítačové experimenty na praktických inštanciách TSP ukazujú, že počet zlepšení štartovacieho riešenia obyčajne neprekročí n, t. j. $|\mathcal{Z}| < n$.

Zo zlepšujúcich heuristík sú najpoužívanejšie rôzne typy *výmenných* heuristík, ktoré hľadajú lacnejšie riešenie jednoduchými výmenami poradia miest v permutácii (obr. 3.9). Realizuje sa potom vždy ten z cyklov, ktorý dáva maximálne zlepšenie.



Obr. 3.9: Výmenné heuristiky (a) krížením a (b) vkladaním

Medzi najmodernejšie metaheuristiky aplikovateľné na úlohu TSP, v ktorých sa môžu uplatiť aj doteraz uvedené heuristiky, patria rôzne druhy *evolučných algoritmov*, ktoré napodobujú cielený vývoj populácie jedincov v prírode pomocou *selekčných* a *variačných* operácií (*mutácia, migrácia a rekombinácia*) jedincov. V roli jedincov tu máme prípustné riešenia TSP, t. j. cyklické permutácie.

Algoritmus 3.12. Evolučný algoritmus pre TSP

Krok 1: Vytvoríme N-člennú inicializačnú populáciu $\mathcal{P} = \{\pi_1, \pi_2, \ldots, \pi_N\}$ cyklických permutácií.

Krok 2: Z populácie $\mathcal{P}$ vyberieme (selekcia) „dobrú“ podmnožinu jedincov $\mathcal{P}'$.

Krok 3: Aplikujeme variačné operácie na jedincov $\mathcal{P}'$, tým vytvoríme novú populáciu $\mathcal{P}''$.

Krok 4: Ak nie je splnené nejaké **STOP kritérium**, položíme $\mathcal{P} = \mathcal{P}''$ a **GOTO Krok 2**. Ináč jedinec $\pi \in \mathcal{P}''$ s minimálnym ohodnotením $c(\pi)$ je suboptimálnym riešením.

Nevýhodou uvedených metód je, že nepoznáme kvalitu takto získaných suboptimálnych riešení. Aj keď existujú aproximačné metódy[1] zabezpečujúce daný aproximačný pomer medzi získaným heuristickým riešením a neznámym optimálnym riešením úlohy TSP, nenašli tieto prístupy úspešné uplatnenie v ich zovšeobecneniach.

[1]Metóda kostry a párenia poskytuje aproximačné riešenie, ktorého ocenenie je v najhoršom prípade 3/2 násobkom optimálneho riešenia a je doteraz najlepším známym výsledkom.

Existujú však aj také prípady matíc **C**, pre ktoré je úloha TSP riešiteľná v polynomiálnom čase. Medzi najznámejšie patrí *Mongeho matica*, t. j. taká matica, v ktorej pre všetky indexy $i, j, k, l \in \mathcal{N}$ také, že pre $i < k$ a $j < l$ platí

$$c_{i,j} + c_{k,l} \leq c_{i,l} + c_{k,j}.$$

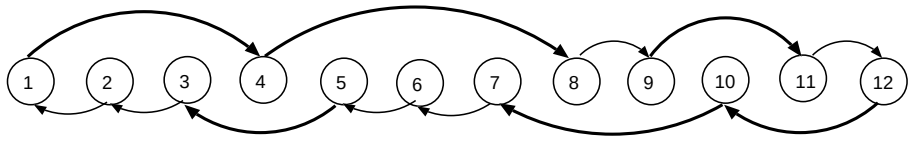
V monografii [30] sa uvádza, že cyklus $1 \to 3 \to 5 \to \cdots \to 6 \to 4 \to 2$ je riešením úlohy so symetrickou Mongeho maticou. Na charakteristiku optimálneho riešenia úloh TSP s asymetrickými Mongeho maticami treba koncepciu *pyramídového cyklu* v tvare

$$1 \to i_1 \to i_2 \to \cdots \to i_r \to n \to j_1 \to j_2 \to \cdots \to j_{n-r-2} \to 1 \qquad (3.22)$$

kde $i_1 < i_2 < \cdots < i_r$ a $j_1 > j_2 > \cdots > j_{n-r-2}$. Úloha TSP zúžená na triedu matíc sa nazýva *pyramídovo riešiteľná*, ak pre ľubovoľnú maticu tejto triedy existuje optimálny pyramídový cyklus. Zatiaľčo počet pyramídových cyklov s n mestami je exponenciálny k n, najkratší pyramídový cyklus možno nájsť pomocou dynamického programovania v čase úmernom n^2.

My však uvedieme princíp *grafového pyramídového algoritmu* navrhnutého Peškom [38] s rovnakou zložitosťou, ktorý sa ukázal pri opakovanom použití akceptovateľný ako heuristická metóda aj pre matice bez obmedzení.

Príklad najlacnejšieho pyramídového cyklu $\mathcal{C}$ v úplnom ohodnotenom digrafe G s 12 vrcholmi $1 \to 4 \to 8 \to 9 \to 11 \to 12 \to 10 \to 7 \to 6 \to 5 \to 3 \to 2 \to 1$ je na obr. 3.10. Je konštruovateľný z najlacnejšieho pyramídového 1–12 sledu $\mathcal{S}$



Obr. 3.10: Pyramídový cyklus $\mathcal{C}$ v digrafe G

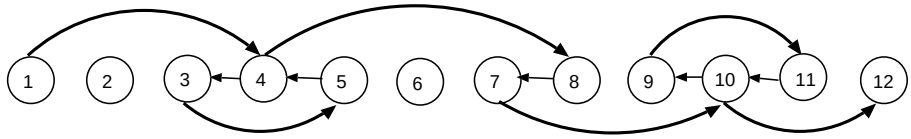
v digrafe G na obr. 3.11. Cyklus $\mathcal{C}$ a sled $\mathcal{S}$ majú súhlasne orientované hrany $(1,4), (4,8), (9,11)$, orientovaným hranám $(5,3), (10,7), (12,10)$ cyklu zodpovedajú opačne orientované hrany $(3,5), (7,10), (10,12)$ sledu a ostatné nezvýraznené hrany cyklu i sledu sú rôzne.

Aby sme dosiahli rovnaké ocenenie cyklu i sledu, potrebujeme, aby hrany $(i, i+1), (i+1, i)$ mali ohodnotenie 0. To môžeme vždy docieliť známou transformáciou $c'_{ij} = c_{ij} - \alpha_i - \beta_j$, ktorá nemá vplyv na optimalitu riešenia. Stačí

postupne položiť

$$\alpha_i = \begin{cases} 0 & \text{ak } i = 1 \\ c_{21} & \text{ak } i = 2 \\ c_{i,i-1} - \beta_{i-1} & \text{ak } 3 \leq i \leq n \end{cases} \qquad \beta_i = \begin{cases} 0 & \text{ak } i = 1 \\ c_{12} & \text{ak } i = 2 \\ c_{i-1,i} - \alpha_{i-1} & \text{ak } 3 \leq i \leq n \end{cases}$$

Problematickými však naďalej zostávajú nesúhlasne orientované hrany cyklu a sledu, ktoré po tejto transformácii môžu mať rôzne ohodnotenie hrán aj pre pôvodne symetrickú maticu ocenení $\mathbf{C}$.



Obr. 3.11: Pyramídový sled $\mathcal{S}$ v digrafe G

V pyramídovom slede $\mathbb{S}$ sa postupne striedajú súhlasne a nesúhlasné orientované hrany digrafu G, čo možno modelovať pomocou acyklického digrafu $\mathcal{G}(G)$ priradenému k digrafu G rozdelením vnútorných vrcholov $i, 1 < i < n$ digrafu G na dvojice vrcholov u_i, v_i ako vidieť z obr. 3.12.

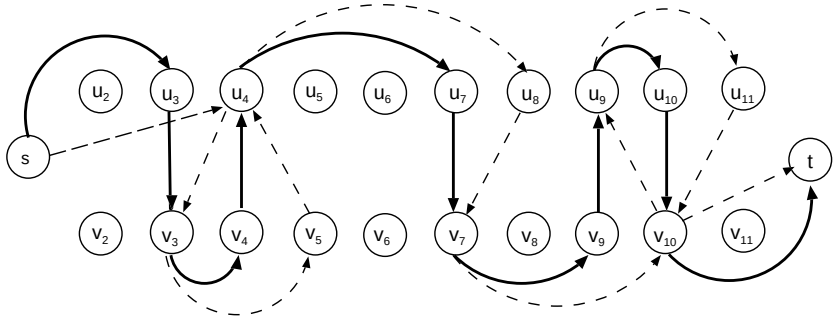
Potom hranám incidujúcim s vrcholmi u_i zodpovedajú súhlasne orientované hrany a hranám incidujúcim s vrcholmi v_i zodpovedajú nesúhlasne orientované hrany. Poznamenajme, že hrana (u_i, u_j) má ohodnotenie $c'_{i,j+1}$, hrana (v_i, v_j) ohodnotenie $c'_{j+1,i}$ a hrany $(u_i, v_i), (v_i, u_i)$ majú ohodnotenie 0. A tak je najlacnejšiemu pyramídovému sledu $\mathcal{S}$ zobrazenému čiarkovane jednoznačne priradená najlacnejšia 1–12 cesta v acyklickom digrafe $\mathcal{G}(G)$ zobrazená hrubými šípkami.

Najlacnejší pyramídový 1–n cyklus tak môžeme hľadať ako najlacnejšiu 1–n cestu v digrafe $\mathcal{G}(G)$. Vidíme, že ohodnotenie hrán v digrafe $\mathcal{G}(G)$ závisí od poradia vrcholov v množine vrcholov V pôvodného digrafu G, t. j. indexov matice $\mathbf{C}$. Túto skutočnosť využíva heuristická metóda, ktorá opakovane hľadá najlacnejší pyramídový cyklus.

Algoritmus 3.13. Algoritmus najlacnejšieho pyramídového cyklu.

Krok 1: Nájdeme cyklická permutáciu π zodpovedajúcu najlacnejšiemu pyramídovému cyklu $1 \to \pi(1) \to \cdots \to 1$ s maticou ocenení $\mathbf{C}$.

Krok 2: Označíme $\mathbf{C}^\pi = c_{\pi(i)\pi(j)}$ maticu, ktorá vznikne súhlasnou permutáciou riadkov a stĺpcov matice $\mathbf{C}$.



Obr. 3.12: s–t cesta pre pyramídový 1–12 sled v acyklickom digrafe $\mathcal{G}(G)$

Krok 3: Nájdeme cyklickú permutáciu ψ zodpovedajúcu najlacnejšiemu pyramídovému cyklu s maticou $\mathbf{C}^{\pi}$.

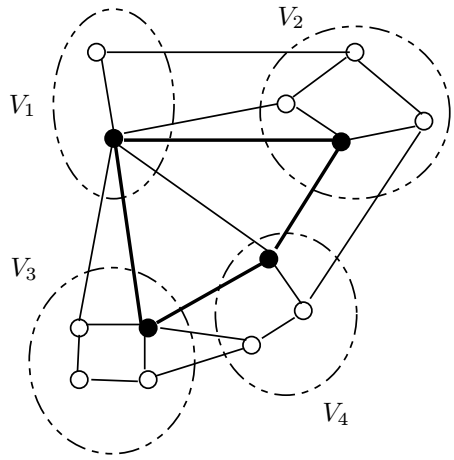
Krok 4: Ak $c(\pi) \leq c^{\pi}(\psi)$, potom **STOP** so suboptimálnym riešením π. Ináč položíme $\pi = \phi(\pi)$ a **GOTO Krok 2**.

Počítačové experimenty ukázali, že metóda je závislá od počiatočného riešenia v kroku 1. Tento jav možno potlačiť náhodnou (súhlasnou) permutáciou pôvodnej matice $\mathbf{C}$ a n krát opakovanou aplikáciou pyramídovej heuristiky. Podarilo sa tak nepatrne (na 6526) zlepšiť doteraz najlepšie (6528) publikované heuristické riešenie Euklidovskej inštancie s $n = 150$.

Medzi zovšeobecnenia úlohy TSP patrí *skupinová úloha obchodného cestujúceho* [47] - (GTSP) Group Traveling Salesman Problem. Je motivovaná skutočnosťou, že pokiaľ sú vozidlom obsluhované vrcholy dopravnej siete zoskupené do dostatočne malých rajónov, potom stačí, ak je navštívený aspoň jeden vrchol každého rajónu. Na obrázku 3.13 máme príklad jazdy vozidla, keď každý zo štyroch regiónov je navštívený práve raz, vybrané vrcholy sú zvýraznené.

Uvažujme jej grafovú formuláciu: Je daný hranovo ohodnotený graf $G = (V, H, d)$, $d : H \to \mathbb{R}_0^+$ a rozklad jej množiny vrcholov V do m tried, t. j. množiny $V_1, V_2, \ldots, V_m$ také, že $V = \bigcup_{k=1}^{m} V_k$ a $V_i \cap V_j = \emptyset$ pre $1 \leq i < j \leq m$. Hľadá sa najlacnejšia cyklická permutácia ψ nad výbermi z tried rozkladov vrcholov, pričom z každej triedy rozkladu možno vybrať najmenej jeden vrchol. Množina všetkých prípustných výberov je tvorená podmnožinami vrcholov, t. j. $\mathcal{S} = \{M \mid M \subset V,\ |M \cap V_k| \geq 1,\ k = 1, 2, \ldots, m\}$.

Pokiaľ sa požaduje výber práve jedného vrcholu z každej triedy rozkladu, hovoríme o *prieberčivej úlohe obchodného cestujúceho – 1STSP – One of Set*



Obr. 3.13: Prípustná jazda vozidla v úlohe GTSP

Traveling Salesman Problem. Na obrázku 3.13 tak máme aj prípustné riešenie úlohy 1STSP. Ako prvý túto úlohu u nás formuloval S. Palúch, niektoré výsledky o nej sú v práci [44] jeho doktoranda M. Pončáka.

Úlohu 1STSP možno formulovať pri označení $V = \{1, 2, \ldots, n\}$ ako nasledujúcu úlohu bivalentného programovania (U1STSP): Nájsť také x_{ij} aby

$$\sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \ \rightarrow \ \min \tag{3.23}$$

za podmienok:

$$\sum_{i \in V} x_{ij} = 1 \qquad j \in V \tag{3.24}$$

$$\sum_{j \in V} x_{ij} = 1 \qquad i \in V \tag{3.25}$$

$$\sum_{i \in V_k} x_{ii} = |V_k| - 1 \qquad k = 1, \ldots, m \tag{3.26}$$

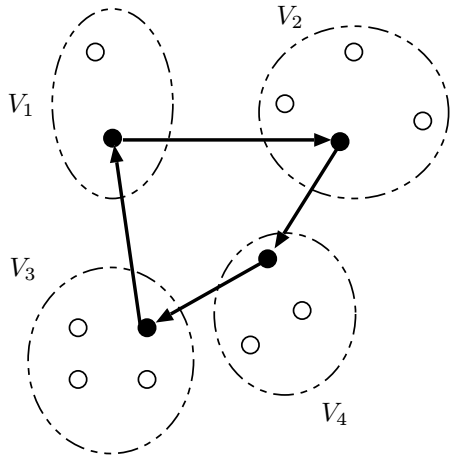
$$\sum_{(i,j) \in P \times P : \{i,j\} \in H} x_{ij} \leq |P| - 1 \quad \forall P \subset S \ \in \mathcal{S}, |P| \geq 2 \tag{3.27}$$

$$x_{ij} \in \{0, 1\} \qquad i \in V, j \in V \tag{3.28}$$

kde symetrická matica $\mathbf{C}$ je definovaná vzťahom

$$c_{ij} = \begin{cases} d(h) & \text{ak } h = \{i,j\} \in H \\ 0 & \text{ak } i = j \\ \infty & \text{ináč} \end{cases} \tag{3.29}$$

Rozhodovacie premenné $x_{ij} = 1$ s rôznymi indexami definujú $\psi(i) = j$ (vybranú hranu z vrcholu i do vrcholu j) a premenné $x_{ii} = 1$ indikujú, že vrchol i nebol vybraný do riešenia. Rovnice (3.24), (3.25) a (3.27) definujú prípustné riešenie priraďovacieho problému. Podmienka (3.26) zabezpečí, aby jediný cyklus dĺžky m reprezentoval hľadanú permutáciu ψ definovanú na množine $S = \{i \mid i \in V,\ x_{ii} = 0\}$. V prípade úlohy GTSP stačí nahradiť rovnicu (3.26) podmienkou $\sum_{i \in V_k} x_{ii} \leq |V_k| - 1$.



Obr. 3.14: Riešenie ψ zodpovedajúce riešeniu úlohy U1STSP

Na obrázku obr. 3.14 máme riešenie úlohy U1STSP zodpovedajúce ilustračnému príkladu na obr. 3.13.

Základná okružná dopravná úloha BVRP (Basic Vehicle Routing Problem) sa často formuluje ako *rozvozná úloha* takto: Je daná množina n zákazníkov a sklad. Je známa matica vzdialeností medzi nimi $\mathbf{D} \in \mathbb{R}^{(m+1)\times(m+1)}$, kde index $i = 0$ zodpovedá miestu skladu a index $i > 0$ miestu zákazníka. K dispozícii je konečný vozidlový park zameniteľných vozidiel danej kapacity Q. Každý zákazník požaduje dovoz daného množstva substrátu b_i zo skladu práve jedným

vozidlom. Cieľom nájsť pre daný počet m vozidiel *rozvozné (okružné) trasy vozidiel* pri minimálnej celkovej prejdenej vzdialenosti vozidiel. Rozvozná trasa vozidla začína v sklade, pokračuje postupnými obsluhami požiadaviek zákazníkov a je ukončená v sklade.

Predpokladajme, že máme k dispozícii všetkých $\mathcal{K}$ prípustných trás vozidiel. Nech M_k je množina indexov zákazníkov na $k-$tej trase vozidla ohodnotenej číslom $d(M_k)$, ktoré je rovné cene optimálneho riešenia TSP v úplnom grafe indukovanom množinou vrcholov $V_k = \{0\} \cup M_k$ a reprezentovanom cyklom C_k. Ak teda $i_1, i_2, \ldots i_r \in M_k$, potom $b_{i_1} + b_{i_2} + \ldots b_{i_k} \leq Q$.

Úlohu BVRP formulovali Christofides a kol. [30] ako úlohu bivalentného programovania (CRP): Nájsť také y_k aby

$$\sum_{k \in \mathcal{K}} d(M_k).y_k \ \rightarrow \ \min \tag{3.30}$$

za predpokladov:

$$\sum_{k \in K; i \in M_k} y_k = 1 \qquad i = 1, 2, \ldots, n \tag{3.31}$$

$$\sum_{k \in \mathcal{K}} y_k = m \tag{3.32}$$

$$y_k \in \{0, 1\} \qquad k \ \in \mathcal{K} \tag{3.33}$$

Rozhodovacia premenná $y_k = 1$, ak bude vybraný k-ty prípustný cyklus C_k a $y_k = 0$ v opačnom prípade. Obmedzenie (3.31) zaručuje, že každý zákazník bude obslúžený v niektorej rozvoznej trase. Podmienka (3.32) zasa zabezpečí, že každému z vozidiel je priradená práve jedna rozvozná trasa vozidla. Cieľová funkcia (3.30) udáva celkovú vzdialenosť prejdenú vozidlami.

Optimalizačná úloha CRP je efektívna, ak je známa (nie príliš početná) množina všetkých prípustných cyklov. Takáto situácia bola reálna, pri rozvoze uhlia v regióne Banská Bystrica boli rozvozné trasy tvorené len nanajvýš tromi zákazníkmi.

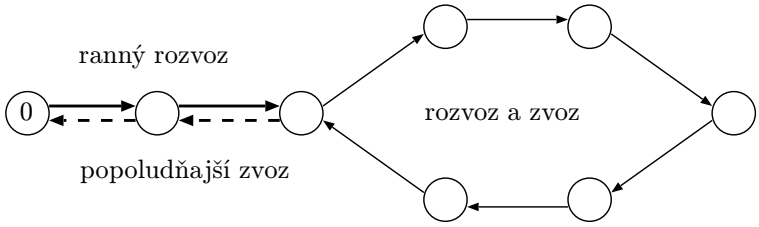
Častejšie však nemáme, vzhľadom na početnosť množiny $\mathcal{K}$, k dispozícii všetky jej prípustné trasy. Potom množina $\mathcal{K}$ obsahuje len indexy perspektívnych rozvozných trás. To môže viesť aj k situácii, že optimalizačná úloha nemá prípustné riešenie. Po doplnení rozvoznými trasami (cyklami) $0 \rightarrow i \rightarrow 0$ má už úloha CRP vždy prípustné riešenie.

V prípade, keď je k dispozícii dostatočne početný vozidlový park alebo vozidlá sú schopné vykonať všetky rozvozné trasy, môžeme upustiť od podmienky (3.32) a dostávame klasickú *úlohu o rozklade* určenú (3.30),(3.31) a (3.33). Ak naviac potrebujeme nájsť najlacnejšiu množinu rozvozných trás pokrytú s minimálnym počtom vozidiel stačí modifikovať cieľovú funkciu $\sum_{k \in \mathcal{K}} \left(d(M_k) +\right.$

M).y_k s dostatočne veľkou penalizačnou konštantou M. Keď ale potrebujeme len zistiť minimálny počet vozidiel bez ohľadu na prejdenú vzdialenosť, stačí použiť cieľovú funkciu $\sum_{k\in\mathcal{K}} y_k$.

Niekedy je výhodou formulácie úlohy CRP, že vychádza z danej množiny prípustných trás vozidla. To znamená, že nám umožňuje riešiť aj *úlohy s časovými oknami*, keď sa nakládka a vykládka musí vykonať v časovom okne (intervale) [37]. V takomto prípade dostávame spravidla len heuristické riešenie. Osvedčilo sa nám zaraďovať do $\mathcal{K}$ už existujúce prevádzkované trasy, trasy navrhnuté dispečermi a trasy navrhnuté rôznymi matematickými modelmi. Pri súčasnej dostupnosti voľne šíriteľného softvéru na riešenie úloh bivalentného programovania sa nám tak podarilo riešiť úlohy s $|\mathcal{K}| \leq 200, m \leq 50$.

Aplikovateľné výsledky sme dosiahli už s metódou perturbačnej schémy kombinovanou s tvoriacou heuristikou Clarka a Wrighta pre BVRP, ktoré nám vytvárali variantné riešenia *zvozu domového odpadu v Žiline*. Podobne sme postupovali aj pri riešení *problému jázd pojazdných predajní* v regióne Rimavská Sobota [37], kde je požiadavka vytvárať pre obyvateľov odľahlých dedín čo najväčšie časové okná predaja, keď vozidlo stojí.

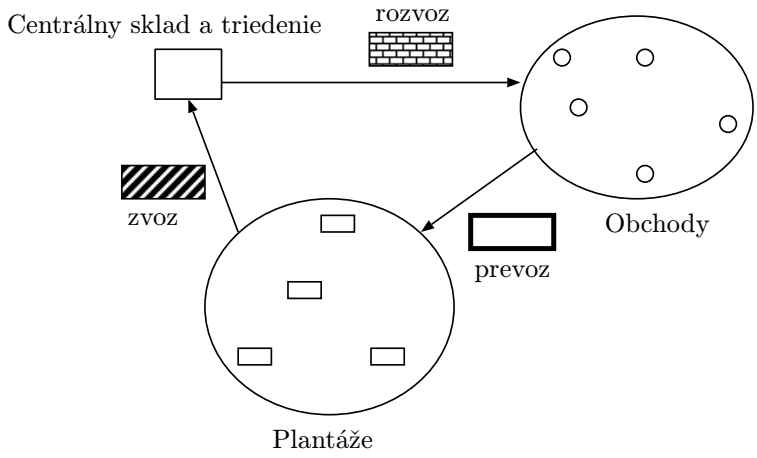


Obr. 3.15: Tri fázy obsluhy dlhej trasy vozidla

Pri *rozvoze mlieka* tiež v regióne Rimavská Sobota a *rozvoze piva* pre ČSAD Bratislava [38] sme sa stretli pri dlhých jazdách vozidiel s potrebou tvorby trojfázových rozvozno-zvozných trás znázornených na obrázku 3.15, kde sa rozvoz týka plných prepraviek a zvoz prázdnych prepraviek. Inak nebolo možné vyhovieť časovým oknám pre vykládku plných a nakládku prázdnych prepraviek v predajniach.

Domnievame sa však, že spolu s riešením príslušných úloh CRP by bolo možné získať ešte kvalitnejšie riešenia.

Model *zvoz-rozvoz-prevoz* schematicky znázornený na obr. 3.16 bol navrhnutý v práci Černého [7], kde sa z plantáží zvážajú plné prepravky tovaru (ovocie, zelenina) do centrálneho skladu, odkiaľ sa plné prepravky triedeného tovaru roz-



Obr. 3.16: Zvoz-rozvoz-prevoz

vážajú do obchodov a následne prázdne prepravky prevážajú späť na plantáže. I tu by bolo možné riešenie pomocou úlohy CRP, pokiaľ by sa našiel efektívny algoritmus tvorby prípustných trás „sklad - obchody - plantáže - sklad“.

3.7 Úlohy o obsluhe hrán dopravnej siete

V okružných dopravných úlohách sú zákazníci umiestnení vo vrcholoch dopravnej siete, kde sa vykonáva ich obsluha. V prípade, keď sa obsluha týka hrán dopravnej siete (napr. pri letnom kropení ulíc, zimnej údržbe ciest, zvoze domového odpadu umiestneného v kontajneroch v blízkosti ciest) vzniká potreba riešiť *úlohy o obsluhe hrán siete*. Ich podrobnú klasifikáciu spolu s množstvom odkazov na literatúru možno nájsť v [1].

Najjednoduchšia formulácia úlohy tohto typu je známa ako *úloha čínskeho poštára* - CPP (Chinese Postman Problem). Verbálne ju možno formulovať takto: Poštár, ktorý začína a končí svoju prácu na pošte, má prejsť všetkými ulicami svojho rajónu tak, aby sa čo najmenej nachodil. Prvá formulácia úlohy je podľa [43] z roku 1962 od čínskeho matematika Kwana, odkiaľ pochádza aj jej názov. Matematická formulácia úlohy v pojmoch teórie grafov a s dopravnou sieťou modelovanou súvislým grafom je nasledujúca: Je daný hranovo ohodnotený

graf $G = (V, H, c),\ c : H \to \mathbb{R}_0^+$. Hľadá sa eulerovský sled (uzavretý sled S grafu obsahujúci každú hranu aspoň raz) s minimálnou dĺžkou $c(S) = \sum_{h \in S} c(h)$.

Exaktné a efektívne (polynomiálne) riešenie úlohy CPP navrhol Edmons (1973). Toto riešenie využíva nasledujúce pojmy párenia a najlacnejšieho úplného párenia v grafe. *Párenie v grafe* G je taký podgraf grafu G, v ktorom má každý vrchol stupeň 1. *Úplné párenie v grafe* G je také párenie v grafe G, ktorého množina vrcholov obsahuje všetky vrcholy grafu G. Cena párenia v hranovohodnotenom grafe je súčet ohodnotení všetkých hrán párenia. *Najlacnejšie úplné párenie* v hranovo ohodnotenom grafe G je úplné párenie v G s minimálnou cenou.

Algoritmus 3.14. Edmonsov algoritmus

Krok 1: Ak je graf G eulerovský (stupne všetkých vrcholov sú párne), položíme $G_E = G$ a **GOTO Krok 4**, ináč definujeme množinu V_E vrcholov grafu G nepárneho stupňa.

Krok 2: Definujme úplný graf $K_{2t} = (V_E, H_E, d)$, kde hranové ohodnotenie každej hrany $\{v_i, v_j\} \in H_E$ je označené d_{ij} a udáva dĺžku najkratšej cesty z vrcholu v_i do vrcholu v_j.

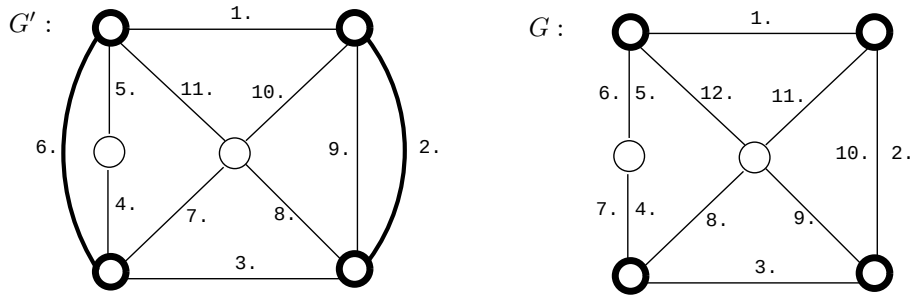
Krok 3: V grafe K_{2t} sa hľadá najlacnejšie úplné párenie $P = (V_E, H_p, d)$, t. j. taký podgraf grafu K_{2t}, ktorého každý vrchol $u \in V_E$ je vrcholom práve jednej páriacej hrany $\{u, v\} \in H_p$ a súčet ohodnotení jeho hrán je minimálny.

Krok 4: K hranám H grafu G pridáme hrany párenia H_p a dostaneme eulerovský (multi)graf $G_E = (V_E, H \cup H_p, c)$, ktorého stupne všetkých vrcholov sú párne - no niektoré hrany sa môžu opakovať.

Krok 5: V grafe G_E sa hľadá eulerovský ťah T_E (uzavretý sled obsahujúci každú hranu grafu práve raz), z ktorého zostrojíme hľadaný uzavretý sled S nahradením páriacich hrán $\{u, v\} \in T_E$ hranami najkratšej cesty z u do v.

Na obr. 3.17 máme príklad grafu G, z ktorého vznikol multigraf G' pridaním dvoch hrán párenia medzi hrubo vyznačenými vrcholmi. Tieto 4 vrcholy majú v grafe G nepárny stupeň 3. Poradové čísla hrán multigrafu G' udávajú poradie hrán v eulerovskom slede T_E. Poradové čísla hrán grafu G udávajú poradie hrán v hľadanom eulerovskom slede S grafu G.

Pri riešení praktických úloh by mohol byť problematický **Krok 3** – riešenie úlohy najlacnejšieho úplného párenia, pre ktorý je síce známa grafová *maďarská*



Obr. 3.17: Eulerovský ťah T_E v multigrafe G'
a príslušný eulerovský sled S grafu G

metóda pracujúca v polynomiálnom čase, no je programátorsky náročná. Pre úlohy stredných rozmerov, kde $|V_E| \leq 50$, máme dobré praktické skúsenosti s nasledujúcou bivalentnou formuláciou úlohy najlacnejšieho úplného párenia (MFM): Nájsť také x_{ij} aby

$$\sum_{i,j;i<j,\{v_i,v_j\}\in H_E} d_{ij}x_{ij} \;\rightarrow\; \min \tag{3.34}$$

za podmienok:

$$\sum_{v_i\in V_E-\{v_j\}} x_{ij} = 1 \qquad v_j \in V_E \tag{3.35}$$

$$\sum_{v_j\in V_E-\{v_i\}} x_{ij} = 1 \qquad v_i \in V_E \tag{3.36}$$

$$x_{ij} - x_{ji} = 0 \qquad \{v_i, v_j\} \in H_E \tag{3.37}$$

$$x_{ij} \in \{0,1\} \qquad v_i \in V_E, v_j \in V_E \tag{3.38}$$

Úloha MFM bez podmienky (3.37) je priraďovacou úlohou, v ktorej sa nepripúšťajú priradenia $v_i \rightarrow v_i$. Podmienka (3.37) nám zabezpečí, že ak $x_{ij} = 1$, $v_i \rightarrow v_j$, potom aj $x_{ji} = 1$, $v_j \rightarrow v_i$ a $\{v_i, v_j\}$ je hranou párenia. A tak pre hľadané páriace hrany máme $H_p = \big\{\{v_i, v_j\} \mid \{v_i, v_j\} \in H_E,\ x_{ij} = 1,\ i < j\big\}$.

Pri hľadaní eulerovského ťahu T_E v eulerovskom multigrafe G_E možno výhodne použiť nasledujúci *labyrintový algoritmus* [43], ktorý je špeciálnym prípadom Tarryho algoritmu pre prieskum labyrintu:

Algoritmus 3.15. Labyrintový algoritmus pre eulerovský ťah

Krok 1: Hľadá sa eulerovský sled S_E (sled obsahujúci každú hranu multigrafu G_E aspoň raz) začínajúci vo vybranom vrchole v_0, pričom sa zaznamenáva smer prechodu po hranách, na základe nasledujúceho pravidla:

Každú hranu možno použiť v jednom smere iba raz, pričom uprednostňujeme hrany v takomto poradí: najskôr nepoužité hrany, potom hrany použité raz, ak nie sú hranami prvého príchodu, a nakoniec hrany prvého príchodu.

Krok 2: Prechádza sa postupne hranami S_E. Hrany použité po druhýkrát v slede S_E tvoria eulerovský ťah T_E.

V prípade, keď je potrebné uvažovať aj s orientáciou hrán siete (napr. ľavá a pravá strana cesty), treba riešiť *orientovanú úlohu čínskeho poštára* - DCPP (Directed Chinese Postman Problem). Potom dostávame grafovú formuláciu analogickú úlohe CPP: Je daný hranovo ohodnotený silne súvislý digraf $G = (V, H, c)$, $c : H \to \mathbb{R}_0^+$. Hľadá sa orientovaný eulerovský sled S s minimálnou dĺžkou $c(S)$.

Edmons a Johnson v 1973 ukázali, že DCPP úloha je rovnako dobre (v polynomiálnom čase) riešiteľná ako jej neorientovaná verzia CPP pomocou tokových formulácií [43].

Prirodzenou kombináciou úloh CPP a DCPP vzniká *zmiešaná úloha čínskeho poštára* - MCPP (Mixed Chinese Postman Problem), kde sú niektoré hrany orientované a iné neorientované. Úlohu možno formulovať takto: Je daný hranovo ohodnotený silne súvislý migraf $G = (V, H, c)$, $H = H_a \cup H_b$, $c : H \to \mathbb{R}_0^+$ a orientovanými H_a a neorientovanými H_b množinami hrán. Hľadá sa eulerovský zmiešaný sled S (obsahujúci všetky hrany aspoň raz, pričom orientované hrany v požadovanom smere) s minimálnou dĺžkou $c(S)$.

Papadimitriou v 1976 dokázal, že táto kombinovaná úloha už patrí medzi NP-tažké problémy. Christofides a kol. navrhli exaktné riešenie úloh metódou vetiev a hraníc a s Lagrangeovou relaxáciou, čo im umožnilo riešiť úlohy s $7 \leq |V| \leq 50$, $3 \leq |H_a| \leq 85$, $4 \leq |H_b| \leq 39$. Norbert a Pickard v 1996 úspešne riešili úlohy s $10 \leq |V| \leq 169$, $2 \leq |H_a| \leq 2876$, $15 \leq |H_b| \leq 1849$ metódou vetiev a rezov. Raghavachari a Veerasamy navrhli doteraz najúspešnejší 3/2-aproximačný algoritmus. Najúspešnejší heuristický algoritmus je z roku 2000 od Corberan a kol.

Menej známou modifikáciou úlohy CPP je prípad, keď je síce dopravná sieť modelovaná grafom, ale hľadá sa orientovaný sled, ktorého dĺžka závisí od toho, v akom smere sa prechádza hranami siete. Zodpovedá to pochôdzke poštára idúceho ulicami v smere alebo proti smeru vetru. Úlohu má názov *úloha*

veterného poštára - WPP (Windy Postman Problem). Možno ju formulovať takto: Je daný súvislý graf $G = (V, H, c_1, c_2)$ ohodnotený dvoma hranovými ohodnoteniami $c_1 : H \to \mathbb{R}_0^+$ a $c_2 : H \to \mathbb{R}_0^+$. Hrana $\{v_i, v_j\} \in H$ je ohodnotená číslom $c_1(v_i, v_j)$, ak bude použitá v smere $v_i \to v_j$, $i < j$, a číslom $c_2(v_i, v_j)$ v opačnom prípade. Opäť sa hľadá eulerovský sled S s minimálnou dĺžkou $c(S) = \sum_{\{v_i,v_j\}\in S, i<j} c_1(v_i, v_j) + \sum_{\{v_i,v_j\}\in S, i>j} c_2(v_i, v_j)$.

Formulácia tohto NP-ťažkého problému pochádza od Minieku z roku 1978. Úlohu môžeme riešiť v dvoch základných krokoch, kde sa v prvom hľadá (v nepolynomiálnom čase) optimálne smerovanie pochôdzok hranami grafu G a v druhom sa konštruuje (v polynomiálnom čase) eulerovský ťah v takto vytvorených orientovaných hranách - môžu byť orientované aj v oboch smeroch.

Algoritmus 3.16. Algoritmus pre WPP

Krok 1: V grafe $G = (V, H, c_1, c_2)$ sa hľadá optimálne riešenie nasledujúcej úlohy (Windy Flow Problem) bivalentného programovania (WFP): Nájsť také x_{ij} aby

$$\sum_{i,j;i<j,\{v_i,v_j\}\in H} c_1(v_i, v_j)x_{ij} + c_2(v_i, v_j)x_{ji} \;\to\; \min \tag{3.39}$$

$$\sum_{i,j;i<j,\{v_i,v_j\}\in H} x_{ij} - x_{ji} = 0 \qquad v_j \in V \tag{3.40}$$

$$x_{ij} + x_{ji} \geq 1 \qquad \{v_i, v_j\} \in H \tag{3.41}$$

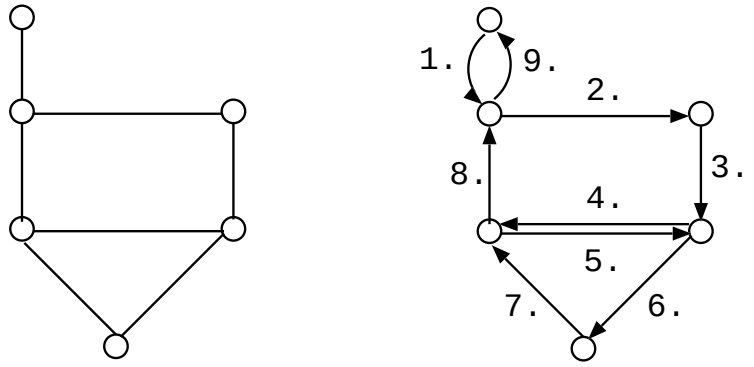
$$x_{ij}, x_{ji} \in \{0, 1\} \quad \{v_i, v_j\} \in H \tag{3.42}$$

Krok 2: V eulerovskom digrafe G_X generovanom riešením $\mathbf{X}$ úlohy WFP, t. j. $G_X = (V, H_X)$, kde

$$H_X = \{(v_i, v_j) \mid (v_i, v_j) \in V \times V,\ x_{ij} = 1,\ \{v_i, v_j\} \in H\},$$

sa hľadá (orientovaný) eulerovský ťah S_X, ktorý zodpovedá hľadanej pochôdzke (eulerovskému sledu) veterného poštára v grafe G.

Exaktné riešenie založené na vhodne definovaných rezných nadrovinách navrhli v roku 1992 Grötschel a Win. Úspešne tak vyriešili úlohy s $52 \leq |V| \leq 264$, $78 \leq |H| \leq 489$. Doteraz najlepší je 2-aproximačný polynomiálny algoritmus pre siete modelované eulerovskými grafmi.



G G_X

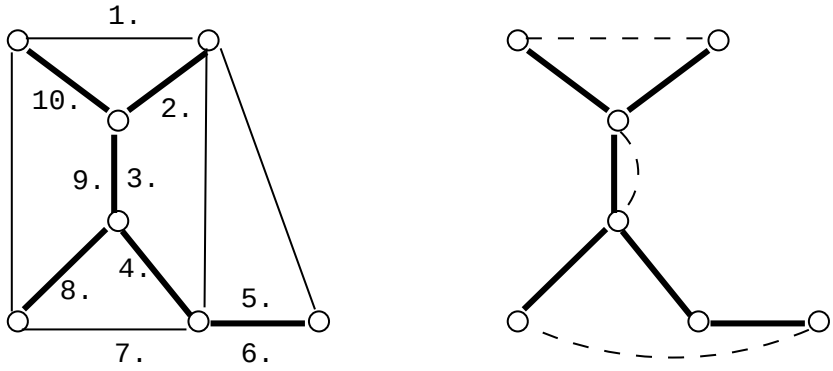
Obr. 3.18: Graf G a príslušný eulerovský digraf G_X a eulerovský ťah S_X

Benavent a kol. (2003) aplikovali evolučný Scatter Search algoritmus a dostali dobré heuristické riešenia pre úlohy s $|V| \leq 988, |H| \leq 3952$.

Formulácie úloh CPP, DCPP, MCPP možno zosiliť požiadavkou, aby boli obsluhované len niektoré hrany $R \subset H$ dopravnej siete. Dostávame tak príslušné verzie RPP, DRPP, MRPP tzv. *úloh dedinských poštárov* (Rural Postman Problem).

V prípade úlohy RPP ide o polynomiálne problémy (dobre riešiteľné) len ak obsluhované hrany R vytvárajú súvislý podgraf dopravnej siete. Na obr. 3.19 máme príklad takých hrán nakreslených hrubou čiarou a príslušné riešenie, ktoré neobsahuje všetky hrany dopravnej siete. Hrany párenia sú zobrazené čiarkovane. Potom totiž opäť stačí (ako v CPP) hľadať pre uvažovaný súvislý podgraf s množinou hrán R príslušné najlacnejšie úplné párenie medzi vrcholmi nepárneho stupňa. Podobne sa dá postupovať aj v prípade úlohy DRPP.

Úlohu RPP riešil exaktne Christofides a kol. (1981) metódou vetiev a hraníc pre rozmery $9 \leq |V| \leq 84$, $13 \leq |H| \leq 184$, $4 \leq |R| \leq 74$ a v roku 1994 a v roku 1994 dosiahli Sanchis and Corberán výrazné zníženie doby výpočtu metódou vetiev a rezov. Starostlivá implementácia tejto metódy umožnila v roku 2000 Ghianimu a Laportemu riešiť úlohy s $|V| \leq 350$. Úloha DRPP sa ukazuje byť náročnejšia než RPP, Christofides a kol. (1989) uvádza inštancie riešených úloh $13 \leq |V| \leq 80$, $7 \leq |R| \leq 74$) Pre úlohu RMPP je publikovaný výsledok



Obr. 3.19: Hľadaný sled úlohy RPP
v sieti so súvislým podgrafom obsluhovaných hrán

Corberán a kol. (1999) s inštanciami $20 \leq |V| \leq 100$, $55 \leq |H| \leq 350$, $15 \leq |R| \leq 200$.

Prirodzeným zovšeobecnením úloh CPP, DCPP, MCPP, WPP sú ich k násobné verzie kCPP, kDCPP, kMCPP, kWPP, keď sa vopred fixuje počet poštárov a požaduje, aby každá hrana siete bola obsluhovaná aspoň jedným poštárom.

Ďalšia modifikácia sa týka výberu najmenej jednej hrany z daného rozkladu množiny hrán. V prípade takéhoto zovšeobecnenia úlohy CPP dostávame *zovšeobecnenú úlohu čínskeho poštára* – (GCPP) Generalized Chinese Postman Problem. Úlohu GCPP možno formulovať takto: Je daný hranovo ohodnotený graf $G = (V, H, c), c : H \to \mathbb{R}_0^+$ a rozklad množiny hrán do m tried, t. j. $H = \cup_{k=1}^{m} H_k$, $H_i \cap H_j = \emptyset$, $1 \leq i < j \leq m$. Hľadá sa uzavretý sled S grafu obsahujúci aspoň jednu hranu z každej triedy rozkladu s minimálnou dĺžkou $c(S)$.

Doteraz nie je známy exaktný ani aproximatívny algoritmus riešenia. Autori Dror a Haouari (2000) však dokázali, že úloha GCPP je NP-ťažká.

Ak máme k dispozícii vozidlo a je známy dopyt (veľkosti požiadaviek) na hranách dopravnej siete, vznikne po úprave úlohy CPP *kapacitná úloha čínskeho poštára* (CCPP) Capacitated Chinese Postman Problem. Možno ju formulovať takto: Je daný hranovo ohodnotený graf $G = (V, H, c, d)$ s jedným hranovým ohodnotením oceňujúcim dĺžky hrán $c : H \to \mathbb{R}_0^+$ a druhým hranovým ohodnotením udávajúcim dopyt na hranách $d : H \to \mathbb{R}_0^+$. Nech je známa

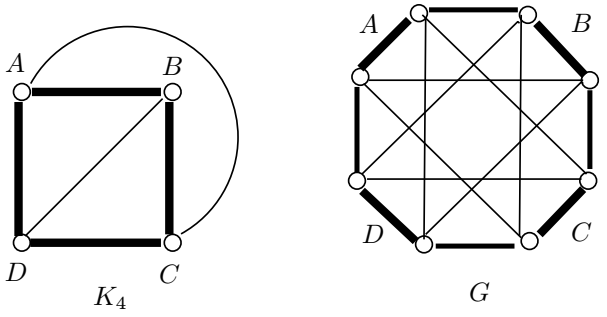
kapacita vozidla Q v jednotkách dopytu a umiestnenie jeho garáže vo vrchole $v_0 \in V$. Potom sa hľadá taká množina uzavretých sledov $\mathcal{S} = \{S_k\}$, ktorá minimalizuje celkovú dĺžku sledov $c(\mathcal{S}) = \sum_{S \in \mathcal{S}} c(S)$ a vyhovuje nasledujúcim podmienkam:

- (P1) Každý sled je uzavretý v_0–v_0 sled; vozidlo začína a končí v garáži
- (P2) Každá hrana je prvkom aspoň jedného sledu; hrana je obslúžená aspoň na jednej trase
- (P3) Dopyt každého sledu $d(\mathcal{S}) = \sum_{S \in \mathcal{S}} d(S)$ je nanajvýš Q; dopyt na trase vozidla neprevýši jeho kapacitu

Poznamenajme v prípade, keď sa podmienka (P2) nahradí podmienkou

- (R2) Každý hrana danej podmnožiny $H' \subset H$ je prvkom aspoň jedného sledu

dostávame úlohu známu ako *kapacitná okružná úloha na hranách* (CARP) Capacitated Arc Routing Problem (CARP). Ľahko sa môžeme presvedčiť, že TSP a BVRP sú špeciálne prípady CARP. Stačí totiž dopyt vo vrcholoch previesť na dopyt na hranách „roztiahnutím vrcholov“, tieto hrany majú nulovej dĺžky a jednotkovú kapacitu. Na obrázku 3.20 zodpovedá cyklu A,B,C,D,A sled



Obr. 3.20: Reprezentácia TSP v K_4
ako sled v sieti G obsahujúci jej jednotkové hrany

v sieti G obsahujúci hrany s jednotkovou kapacitou a nulovou dĺžkou, ostatné hrany majú nulovú kapacitu a dĺžku ako medzi vrcholmi v K_4.

Praktickým problémom zvozu domového odpadu v Liptovskom Mikuláši sa zaoberajú práce I. Stankovianskej [49], [50]. V tomto prípade sa ukázalo efektívnejšie riešiť tento problém ako CARP než ako BVRP.

V prípade, keď je obsluha hrán terminovaná časovou uzávierkou, dostávame napr. z úlohy CPP *úlohu dedinského poštára s uzávierkou* (RPPDC) Rural Postman Problem with Deadline Classes. Tu je daný hranovo a časovo ohodnotený graf $G = (V, H, c, t)$ a nejaký významný vrchol v_0 (garáž-sklad). Je daný rozklad $R_1, R_2, \ldots, R_m$ danej podmnožiny $R \subset H$ množiny hrán, pričom sa požaduje, aby hrany $h \in R_k$ boli obslúžené do daného termínu T_k. Hľadá sa uzavretý v_0–v_0 sled obsahujúci hrany z R, ktoré sú v danom termíne obslúžené, ak sled začína vo vrchole v_0 v čase 0.

Všeobecnejšiu možnosť uprednostňovania obsluhy hrán siete ako RPPDC rieši *hierarchická úloha čínskeho poštára* (HCPP) – Hierarchical Chinese Postman Problem. Je daný hranovo ohodnotený graf $G = (V, H, c)$ a rozklad podmnožiny hrán do tried rozkladu $H_1, H_2, \ldots H_m$ s reláciou precedencie $\prec$ medzi triedami tohto rozkladu. Ak je $H_p \prec H_q$, potom všetky hrany H_p musia byť obslúžené pred hranami H_q. Hľadá sa uzavretý sled minimálnej dĺžky, ktorý rešpektuje reláciu precedencie, pričom obsahuje každú hranu najmenej raz.

Úlohu HCPP formulovali v roku 1987 Dror a kol. pričom dokázali, že je NP-ťažká. Ukázali tiež, že je polynomiálne riešiteľná v prípade, keď je G buď úplný graf alebo úplný digraf, relácia precedencie je lineárne usporiadanie a každá trieda rozkladu vytvára úplný podgraf grafu G.

Exaktné riešenie metódou dynamického programovania navrhol Gélinas v roku 1992. Aproximatívne ani heuristické algoritmy neboli doteraz publikované.

3.8 Lokačné úlohy s daným počtom stredísk

Pri *lokačných* úlohách sa rozhoduje o umiestnení a v niektorých prípadoch i o počte stredísk obsluhy v dopravnej sieti podľa vybraného optimalizačného kritéria. Toto optimalizačné kritérium závisí od účelu a cieľa, ktorý majú umiestňované strediská plniť.

Pri *alokačných úlohách* je množina stredísk obsluhy daná. Pre túto množinu stredísk sa daná sieť rozdeľuje na podsiete (tzv. atrakčné obvody) pridelené na obsluhu jednotlivým strediskám. Ide tu o tzv. problém rajonizácie.

Strediskom obsluhy môže byť napríklad sklad tovaru, prístav, veľká autobusová alebo železničná stanica, letisko, výrobná prevádzka, elektráreň, obchod,

železničné či autobusové depo a podobne. Kriteriálna funkcia pri tomto type stredísk bude vyjadrovať celkové náklady na dopravnú obsluhu siete.

Odlišný charakter majú obslužné miesta typu *havarijného strediska*, ako je požiarna zbrojnica, stanica záchrannej lekárskej služby, strediská na riešenie havárií plynu, vody alebo ekologických havárií. Kvalita rozmiestnenia stredísk obsluhy sa tu meria maximom z časov potrebných na dosiahnutie obsluhovaných uzlov siete.

Od spomenutých dvoch najčastejšie sa vyskytujúcich charakteristík obslužných stredísk máme i strediská s ďalšími vlastnosťami, napríklad skládky nebezpečného odpadu, ktoré sa snažíme umiestňovať čo najďalej od ostatných uzlov siete.

V tejto časti budeme dopravnú sieť modelovať súvislým hranovo a vrcholovo ohodnoteným grafom $G = (V, H, c, w)$, kde $c : H \to \mathbb{R}$, $w : V \to \mathbb{R}$. Ohodnotenie $c(h)$ hrany $h \in H$ predstavuje dĺžku hrany h, ohodnotenia $w(v)$ vrchola $v \in V$ je váha vrchola (predstavuje náročnosť vrchola na obsluhu, resp. relatívnu dôležitosť vrchola v).

Nech $G = (V, H, c, w)$ je dopravná sieť, $D \subseteq V$ podmnožina vrcholovej množiny V, $v \in V$. Potom *vzdialenosť* $d(v, D)$ *vrchola* v *a množiny* D (resp. vzdialenosť $d(D, v)$ množiny D a vrchola v) definujeme nasledujúco:

$$d(v, D) = d(D, v) = \min_{x \in D}\{d(v, x)\}, \tag{3.43}$$

kde $d(v, x)$ je vzdialenosť vrcholov $v,\ x$ v grafe G.

V nasledujúcich častiach uvedieme niektoré lokačné úlohy, v ktorých budeme predpokladať, že máme daný počet p stredísk obsluhy. Pre každú z nich naformulujeme príslušnú kriteriálnu funkciu, pre ktorú budeme hľadať takú p-prvkovú podmnožinu D_p vrcholovej množiny, ktorá ju minimalizuje.

3.8.1 Minimalizácia celkových dopravných nákladov

Nech váha $w(v)$ vrchola $v \in V$ predstavuje množstvo tovaru, ktoré do vrchola v treba dodať za jednotku času. Predpokladajme, že dopravné náklady na obsluhu vrchola v sú priamo úmerné jeho vzdialenosti od najbližšieho depa z D (z ktorého bude vrchol v zásobovaný) a množstvu $w(v)$ - teda dopravné náklady na obsluhu vrchola za jednotku času v budú rovné $k.w(v).d(v, D)$, kde k je vhodná konštanta. Celkové dopravné náklady na obsluhu všetkých vrcholov za jednotku času budú

$$N = k. \sum_{v \in V} w(v).d(v, D),$$

N je teda priamo úmerné súhrnnej váženej vzdialenosti

$$f(D_p) = \sum_{v \in V} w(v).d(v, D) \tag{3.44}$$

všetkých vrcholov grafu G od množiny diep D.

Ak chceme nájsť optimálnu p-prvkovú množinu diep, ktorá minimalizuje celkové dopravné náklady na obsluhu všetkých vrcholov grafu G, stačí nájsť takú p-prvkovú množinu diep D_p, pre ktorú je $f(D_p)$ minimálne. Takúto množinu D_p nazveme *vážený p-medián* grafu G. Špeciálne, ak $w(v) = 1$ pre všetky $v \in V$, hovoríme, že D_p je **p-medián**.

3.8.2 Optimalizácia dostupnosti

Pri tomto prístupe hľadáme také umiestnenie p diep, t. j. takú p-prvkovú množinu $D_p \subseteq V$, pre ktorú je vážená vzdialenosť najhoršie položeného vrchola siete od množiny diep minimálna.

Nech $D_p \subseteq V$ je p-prvková množinu diep. Vzdialenosť najvzdialenejšieho vrchola od tejto množiny diep vypočítame ako

$$\mathrm{ec}(D_p) = \max_{v \in V}\{d(v, D_p)\},$$

a nazveme *excentricita množiny* D_p.
Pre prípady, keď vrcholy dopravnej siete nie sú rovnako dôležité z hľadiska obsluhy, zavádzame pojem *vážená excentricita* $ecc(D_p)$ *množiny* D_p takto:

$$\mathrm{ecc}(D_p) = \max_{v \in V}\{w(v).d(v, D_p)\}. \tag{3.45}$$

Vážená excentricita množiny D_p je vážená vzdialenosť najhoršie položeného vrchola od množiny D_p. Vyjadruje kvalitu množiny havarijných stredísk D_p z hľadiska kvality obsluhy najhoršie položeného vrchola vzhľadom na D_p.

Pri umiestňovaní havarijných stredísk hľadáme také $D_p \subseteq V$, pre ktoré je hodnota $\mathrm{ecc}(D_p)$ minimálna – takéto D_p nazveme *vážené p-centrum* grafu G. Špeciálne ak $w(v) = 1$ pre všetky $v \in V$, (t. j. ak $\mathrm{ecc}(D) = \mathrm{ec}(D)$, hovoríme, že D_p je *p-centrum*.

3.8.3 Ďalšie kriteriálne funkcie

p-maxián

Hľadáme množinu stredísk $D_p \subseteq V$ tak, aby sa maximalizoval súčet vážených vzdialeností medzi obsluhovanými miestami a im najbližšími obslužnými stre-

diskami, t. j. aby hodnota

$$f(D_p) = \sum_{v \in V} w(v).d(v, D)$$

bola maximálna.

Tento prístup možno použiť napríklad pri rozmiestňovaní stredísk s negatívnym dopadom na životné prostredie.

p-anticentrum

Pri tomto prístupe maximalizujeme minimálnu váženú vzdialenosť medzi obsluhovanými miestami a k nim najbližšími obslužnými strediskami. Hľadáme množinu stredísk $D_p \subseteq V$, ktorá maximalizuje kriteriálnu funkciu

$$\mathrm{acc}(D_p) = \min_{v \in V}\{w(v).d(v, D_p)\}.$$

Podobne ako v prípade p-maxiánu je tento prístup použiteľný v prípade rozmiestňovania stredísk s negatívnym dopadom na okolie.

Maximalizácia p-disperzie

Pri tejto rozmiestňovacej úlohe sa maximalizuje funkcia

$$\sigma(D_p) = \min_{v \in D_p} \{w(v).d(v, D_p - \{v\})\}.$$

K najčastejším aplikáciám tejto úlohy patria problémy optimálneho rozmiestnenia stredísk, ktoré sa navzájom negatívne ovplyvňujú. Inou aplikáciou je rozptýlenie zásobníkov (pohonných hmôt, muničných skladov, energetických zdrojov) tak, aby sa minimalizovalo nebezpečenstvo ich súčasného zničenia.

3.8.4 Zámenná heuristika na hľadanie optimálnej p-prvkovej množiny diep s ľubovoľnou kriteriálnou funkciou g

Algoritmus 3.17. Zámenná heuristika

Krok 1. Náhodne vyber p-prvkovú podmnožinu D_p množiny V.
Nech $D_p = \{v_1, v_2, \ldots, v_p\}$, $V - D_p = \{u_1, u_2, \ldots, u_q\}$, kde $q = |V| - p$.

Krok 2. Hľadaj $i,\ j$, $1 \leq i \leq p$, $1 \leq j \leq q$ také,
že pre $D'_p(i,j) := (D_p \cup \{u_j\}) - \{v_i\}$ je $g(D_p) \leq g(D'_p)$.

Krok 3. Ak taká dvojica indexov $i,\ j$ neexistuje STOP.
Inak polož $D_p := D'_p(i,j)$ a GOTO Krok 2.

Algoritmus možno spustiť viackrát s rôznymi štartovacími množinami D_p, po ukončení algoritmu si zapamätať príslušné suboptimálne riešenie a z takýchto suboptimálnych riešení vybrať najlepšie.

Jednou z jednoduchých možností vylepšenia algoritmu je aplikovať pri ňom tzv. perturbačnú schému: dočasne zakázať umiestniť do niektorého vrchola depo a spustiť algoritmus. Ak výsledkom bude lepšie riešenie, ako je doterajší rekord, zákaz urobiť trvalým, inak zákaz zrušiť. V ďalších krokoch najprv dočasne zakázať ďalší vrchol a rozhodnúť o tom, či bude zákaz trvalý atď. Algoritmus sa zastaví, ak nemožno nájsť vrchol, zákazom ktorého sa dosiahne zlepšenie. To môže ale trvať veľmi dlho, preto môžeme algoritmus zastaviť kedykoľvek po dosiahnutí aspoň jedného suboptimálneho riešenia.

Inými možnosťami je aplikácia vyspelých heuristických princípov, ako je tabu search, simulated annealing, genetické algoritmy atď. Aplikácia týchto metód je však nad rámec tejto publikácie.

3.9 Lokačné úlohy o optimalizácii počtu stredísk

3.9.1 Úlohy o pokrytí

Úloha o pokrytí p-mediánom

Táto úloha rieši nasledujúci problém: Na obsluhu územia máme obmedzené prostriedky. Z tohto obmedzenia vyplýva, že $f(D_p) \leq M$. Aký je najmenší možný počet diep p a aké je príslušné rozmiestnenie diep – t. j. aká je príslušná množina D_p – taká, aby $f(D_p) \leq M$? ($f(D_p)$ je definované vzťahom (3.44)).

Úloha o pokrytí p-centrom

Táto úloha je podobná predchádzajúcej úlohe. Rieši problém: Aký minimálny počet diep potrebujeme a aké je príslušné rozmiestnenie diep D_p také, aby vážená vzdialenosť každého vrchola $v \in V$ od D_p bola menšia než zadaná hodnota W? Hľadáme teda také p a príslušné D_p aby $\mathrm{ecc}(D_p) \leq W$, kde $\mathrm{ecc}(D_p)$ je definované vzťahom (3.45).

Obe spomenuté úlohy možno riešiť opakovaným výpočtom p-mediánu, resp. p-centra pre rôzne p zámennou heuristikou s využitím ľahko dokázateľnej vlastnosti, že ak $p < q$, a D_p, resp. D_q je vážený p-medián, resp. vážený q-medián, potom $f(D_p) \geq f(D_q)$.

Kombinované úlohy

Nech P sú náklady na udržiavanie jedného centra za jednotku času, nech dopravné náklady za jednotku času sú $Q.f(D_p)$, kde $D_p \subseteq V$ je p-prvková množina diep. Minimalizovať súčet nákladov za prevádzku p diep a dopravné náklady znamená minimalizovať

$$F(D_p) = P.p + Q.f(D_p) = p.P + Q.\sum_{v \in V} w(v).d(v, D),$$

presnejšie, nájsť p a D_p tak, aby hodnota funkcie $F(D_p)$ bola minimálna.
Podobná úloha pre p-centrum – minimalizovať

$$G(D_p) = P.p + R.\mathrm{ecc}(D_p),$$

kde R je nejaká konštanta a $\mathrm{ecc}(D_p)$ definované vzťahom (3.45) už takú jasnú ekonomickú interpretáciu nemá.

Majme nasledujúcu úlohu. V uzle $i \in V$ možno vybudovať stredisko s maximálnou kapacitou a_i jednotiek tovaru za jednotku času. Nech uzol $j \in V$ potrebuje za jednotku času b_j jednotiek tovaru. Nech f_i sú náklady na prevádzku strediska v uzle i za jednotku času. Nech d_{ij} sú náklady na prevoz jednotkového množstva tovaru od strediska i k uzlu j.

Našou úlohou je určiť, v ktorých uzloch bude zriadené stredisko a tiež určiť pre každý uzol $j \in V$ množstvo tovaru x_{ij}, ktoré mu dodá stredisko i tak, aby celkové náklady na prevádzku stredísk a prepravu tovaru boli minimálne.

Označme y_i rozhodovaciu premennú, pre ktorú platí

$$y_i = \begin{cases} 1 & \text{ak v uzle } i \text{ bude zriadené stredisko} \\ 0 & \text{inak} \end{cases}$$

Potom práve formulovanú úlohu možno zapísať ako nasledujúcu úlohu zmiešaného celočíselného programovania:

$$\text{Minimalizovať} \quad \sum_i f_i y_i + \sum_i \sum_j d_{ij} x_{ij} \tag{3.46}$$

Za predpokladov

$$\sum_{j \in V} x_{ij} \leq y_i . a_i \qquad \text{pre } i \in V \tag{3.47}$$

$$\sum_{i \in V} x_{ij} \geq b_j \qquad \text{pre } j \in V \tag{3.48}$$

$$x_{ij} \geq 0 \qquad \text{pre } i, j \in V \tag{3.49}$$

$$y_i \in \{0, 1\} \qquad \text{pre } i \in V \tag{3.50}$$

Prvá suma v (3.46) predstavuje náklady na prevádzku vybraných stredísk. Druhá suma v (3.46) vyčísľuje dopravné náklady a je totožná s kriteriálnou funkciou dopravnej úlohy. Podmienka (3.47) hovorí, že ak bol uzol i vybratý do množiny stredísk, t. j. ak $y_i = 1$, potom celkové množstvo tovaru dodaného týmto strediskom nesmie presiahnuť jeho kapacitu a_i. Ak uzol i nie je strediskom, t. j. ak $y_i = 0$, potom nedodá nič. Podmienka (3.48) žiada, aby každý odberateľ dostal požadované množstvo. Nerovnosť $\geq$ by v tejto podmienke mohla byť nahradená rovnosťou $=$, druhá suma v kriteriálnej funkcii (3.46) však všade zaistí rovnosť.

Práve formulovaná úloha je NP-ťažká. Pre menšie rozmery ju zvládnu dostupné balíky pre celočíselné a zmiešané lineárne programovanie. Inou možnosťou pre menšie rozmery ju možno riešiť tak, že pre všetky rozumné kombinácie diep sa vypočíta príslušná dopravná úloha a vyberie sa najlepšie riešenie.

Pre väčšie rozmery možno použiť nasledujúci postup: Fixujeme p a pre toto pevné p nájdeme suboptimálne umiestnenie p diep D_p princípom zámennej heuristiky, v ktorej sa ako kriteriálna funkcia $h(D_p)$ vezme optimálna hodnota kriteriálnej funkcie (3.46) úlohy definovanej vzťahmi (3.46) – (3.50), do ktorej dosadíme za $y_i = 1$ pre $i \in D_p$ a $y_i = 0$ inak. Dosadením do (3.46) – (3.50) za y_i dostaneme klasickú dopravnú úlohu. Toto možno postupne robiť pre rôzne p a nájsť tak suboptimálne riešenie danej úlohy.

Pre veľké rozmery lokačných úloh vyvinul J. Janáček náročné heuristické metódy uvedené v [23], ktoré však presahujú rámec tejto publikácie.

Kapitola 4

Zásobovacia a obstarávacia logistika

Zásobami sa rozumejú komodity (suroviny/výrobky) určené na neskoršiu spotrebu. Potreba zladenia *veľkosti zásob* surovín, výrobných zásob a zásob hotových výrobkov s príslušnými informáciami o mieste a čase spotreby patria medzi najstaršie úlohy zásobovacej logistiky. Cieľom snaženia podnikateľa či prenájomcu skladu je nájsť optimálnu stratégiu riešenia tohto konfliktu ponuky a dopytu počas celého sledovaného obdobia.

S prvým pokusom riešiť problém výšky zásob ako optimalizačnú úlohu sa stretávame už v roku 1888, keď sa hľadala optimálna výška pokladničnej hotovosti v peňažnom ústave. Príliš vysoká hotovosť vedie k stratám na úrokoch, zatiaľčo príliš nízka hotovosť môže spôsobiť nedostatok peňazí v pokladni.

Hospodárska depresia 70. rokoch minulého storočia viedla k zhoršeniu hospodárskych výsledkov podnikov. Pri hľadaní nákladových rezerv sa zistilo, že podniky ich majú neprimerane viazané v zásobách. Problém zásob sa začal chápať ako vzťah medzi *čerpaním* a *doplňovaním* zásob v rôznych typoch skladov a skladových sieťach. Ukázalo sa, že efektívne riešenia tu poskytujú klasické optimalizačné a štatistické metódy.

Hospodársky rast v 80. rokoch viedol k individualizácii dopytu. Nároční zákazníci začali požadovať nielen výber, cenu, kvalitu ale aj rýchle dodávky tovaru. Vznikli flexibilné továrne minimalizujúce časové straty rýchlym prestavovaním operácií na výrobných linkách. Následná malosériovosť výroby viedla aj k potrebe primeraných – *kapacitne* menších skladov.

Príchod PC umožnil v reálnom čase sledovať *stav zásob*. Zaujímavým zistením bolo, že hodnototvorný proces trvá len cca 5 % potrebného času, zatiaľ čo 95 % času sa neúčelne tvoria zásoby a prestoje pri manipulácii.

V súčasnosti sa preferencie v skladovaní – aj vďaka globalizácii presunuli na spoľahlivosť a úplnosť dodávky s krátkymi *dodacími lehotami*. Tomuto trendu sa prispôsobujú aj moderné zásobovacie modely orientované na optimalizáciu *zásobovacích reťazcov*.

4.1 Charakteristiky zásobovacích úloh

Príjem skladovaných komodít do skladu budeme nazývať *dodávka* a jeho výdaj *spotreba*. Firmám skladovanie surovín ani tovaru neprináša zisk. Preto vzniká potreba riešiť *optimalizačné úlohy*, ktoré hľadajú *optimálne charakteristiky skladov* tak, aby sa zabezpečil plynulý chod výroby alebo obeh jednotlivých komodít *s minimálnymi celkovými nákladmi* na tvorbu a údržbu zásob pri zvolenom režime práce skladov.

Pre jednoduchosť vyjadrovania sa pri označení jednotiek obmedzíme na v ekonomike sa najčastejšie vyskytujúce jednotky: *1 rok* pre časovú jednotku, *1 Sk* pre menovú jednotku, *1 jedn.* pre jednotku množstva (*1 kus, resp. 1 tona*) skladovanej komodity.

Ďalej budeme uvažovať len jednotkové náklady spojené s režimom práce skladu v tomto členení:

- *akvizičné náklady* C_A $[Sk]$ – súvisia so zabezpečením jednej dodávky do skladu, kumulované náklady rastú priamo úmerne s počtom dodávok,

- *skladovacie náklady* C_S $[Sk/jedn{\cdot}rok]$ – kumulované náklady rastú priamo úmerne so skladovaným množstvom a dobou skladovania, napr. úroky z obchodného úveru kryjúce zásoby alebo náklad stratenej príležitosti ako ušlý úrok z terminovaného vkladu použitého na nákup zásob,

- *aktuálna cena skladovania* C_K $[Sk/jedn.]$ – kumulovaná aktuálna cena sa mení úmerne so skladovaným množstvom komodidty,

- *náklady deficitu* C_D $[Sk/jedn.]$ – vznikajúce pri neuspokojení spotreby chýbajúcou komoditou v sklade v čase príchodu požiadavky na výber zo skladu, kumulované náklady rastú s veľkosťou deficitu[1],

[1] Alternatívnou jednotkou nákladov deficitu by mohla byť $[Sk/jedn \cdot rok]$.

- *náklady (obstarávania) nákupu* C_N $[Sk/jedn.]$ – sú výdaje firmy na nákup komodity do skladu, kumulované výdaje sú rovné jednotkovým nákladom nákupu komodity vynásobenej jej nakupovaným množstvom.

Obmedzíme sa na základné charakteristiky skladov vystupujúce v modeloch ako *deterministické*, resp. *náhodné* veličiny:

- *veľkosť dodávky* Q $[jedn.]$ – najčastejšie požadované je ekonomické množstvo komodít v dodávkach pri konštantnej veľkosti dodávkového cyklu,
- *(ročná) intenzita spotreby* λ $[jedn./rok]$ – udáva priemerný počet spotrebovaných jednotiek zásob za jednotku času,
- *veľkosť poistných zásob* $U[jedn.]$ – zabezpečuje ešte tolerovaný deficit zásob,
- *veľkosť hladiny objednania* $h[jedn.]$ – udávajú stav zásob v čase vystavenia objednávky,
- *veľkosť kapacity skladov* $K[jedn.]$ – obmedzujú veľkosť dodávok a spotreby v dodávkových cykloch,
- *dĺžky dodávkových cyklov* C $[rok]$ – koordinujú dodávky so spotrebou,
- *dĺžky dodacích lehôt* L $[rok]$ – zabezpečujú včasný príjem dodávok.

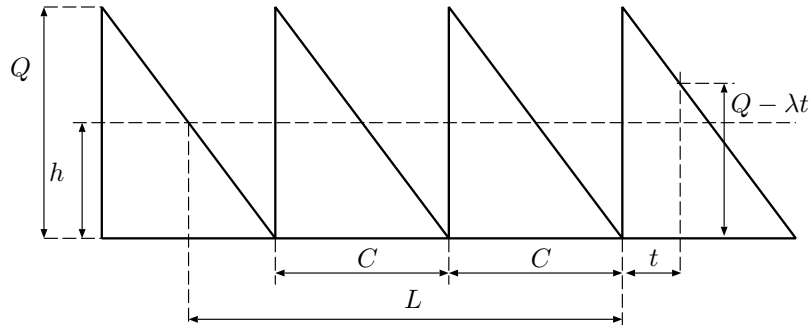
Zásoby sa z hľadiska ich funkcie niekedy delia na *obratové* (zabezpečujúce bežnú potrebu) a *poistné* (zabraňujúce vzniku deficitu zásob). Poistná zásoba sa vytvára najčastejšie s cieľom znížiť vplyv náhodnej spotreby (dopytu). Jej udržiavanie zvyšuje náklady na skladovanie, ale znižuje náklady súvisiace s deficitom zásob. Efektívne riadenie oboch typov zásob predpokladá určenie ich optimálnej veľkosti dodávky, pri ktorom príslušné náklady dosiahnu minimálnu hodnotu a súčasne sú splnené zvolené pravidlá objednávania, skladovania a vyskladňovania zásob.

4.2 Základné modely zásob

Najskôr sa budeme zaoberať najpoužívanejším deterministickým Wilsonovým modelom zásob, ktorý má nielen metodický, ale aj praktický význam. Potom popíšeme niektoré jeho praktické zovšeobecnenia, ktoré vznikajú pri riešení problému deficitu po ocenení deficitu zásob, dodacej lehoty a rabatov dodávky. Stochastickú verziu modelov s deficitom využijeme pri odhade veľkosti poistnej zásoby.

4.2.1 Wilsonove modely ekonomickej veľkosti dodávky

Vývoj stavu zásob v tomto **základnom Wilsonovom modeli** je znázornený na obr. 4.1, vychádza z nasledujúcich predpokladov:



Obr. 4.1: Vývoj stavu zásob v základnom Wilsonovom modeli

- Veľkosť dodávky do skladu Q je neobmedzená a realizuje sa naraz v čase prijatia dodávky.
- Spotreba je lineárnou funkciou času s konštantnou intenzitou λ. Konštanta λ udáva množstvo jednotiek komodity spotrebované za rok.
- Skladovateľnosť komodity – doba pobytu komodity v sklade je časovo neobmedzená.
- Deficit zásob sa nepripúšťa – dodávka príde do skladu práve v čase vyprázdnenia skladu, takže možno z neho plynulo čerpať.
- Sú známe konštantné akvizičné náklady (na jednu dodávku) C_A a konštantné ročné náklady na skladovanie jednotkového množstva komodity za jednotku času C_S.
- Úlohou je určiť *optimálnu veľkosť dodávky* Q^* do skladu, pri ktorej bude *funkcia očakávaných ročných nákladov* (za jednotkové obdobie) minimálna.

Veľkosť dodávkového cyklu C je tu vzhľadom na konštantnú intenzitu spotreby a na nulový deficit tiež konštantná. Okamžitý stav zásob v čase $t \in \langle 0, C)$

je lineárnou funkciou $Q - \lambda t$. Pre $t = C$ máme $Q - \lambda C = 0$, a tak

$$\lambda = \frac{Q}{C} \tag{4.1}$$

Priemerný stav zásob $\bar{Q}$ v jednom dodávkovom cykle C je aritmetický priemer maximálneho a minimálneho stavu zásob na začiatku a konci cyklu

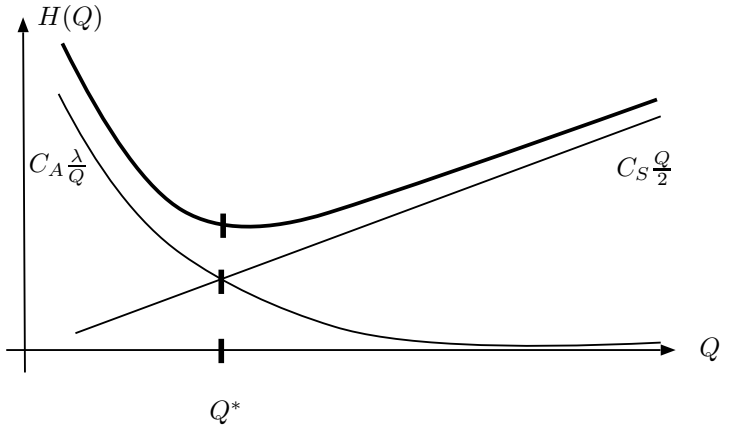
$$\bar{Q} = \frac{1}{C}\int_0^C (Q - \lambda t)dt = Q - \frac{\lambda C}{2} = \frac{Q}{2} \tag{4.2}$$

Obsah každého trojuholníka na obr. 4.1 je tu vyjadrený pomocou integrálu vo vzťahu (4.2), ktorý udáva *veľkosť kumulovaných zásob* v príslušnom dodávkovom cykle. Výpočet priemerného stavu zásob pomocou integrálu je nevyhnutný v modeloch s nekonštantnou spotrebou, napr. pri skladovaní potravín.

S využitím vzťahov (4.1) a (4.2) dostaneme nákladovú funkciu $\mathcal{H}(Q)$, ktorá udáva *priemerné ročné náklady pri veľkosti dodávky* Q, v tvare

$$\mathcal{H}(Q) = (C_A + C_S \cdot C \cdot \bar{Q})\frac{1}{C} = C_A\frac{1}{C} + C_S\bar{Q} = C_A\frac{\lambda}{Q} + C_S\frac{Q}{2} \tag{4.3}$$

Nákladová funkcia je teda súčtom ročných nákladov za dodávky na sklad



Obr. 4.2: Nákladová funkcia $\mathcal{H}(Q)$ v základnom Wilsonovom modeli

a ročných nákladov za skladovanie komodity, obr. 4.2. Je to pre $Q > 0$ rýdzo

konvexná funkcia, a tak nadobúda jediný extrém, ktorý môžeme nájsť pomocou derivácie nákladovej funkcie[2]

$$\frac{d\mathcal{H}(Q)}{dQ} = -C_A\frac{\lambda}{Q^2} + \frac{C_S}{2} = 0, \tag{4.4}$$

odkiaľ dostávame známy *Wilsonov vzorec* pre *ekonomickú (optimálnu) veľkosť dodávky*

$$Q^* = \sqrt{\frac{2\lambda C_A}{C_S}}, \tag{4.5}$$

s ročnou ekonomickou hodnotou nákladov

$$\mathcal{H}(Q^*) = C_A\frac{\lambda}{Q^*} + C_S\frac{Q^*}{2} = \sqrt{2\lambda C_A \cdot C_S}\ .$$

Z vývoja stavu zásob na obr. 4.1 možno vypočítať *hladinu objednania* h, stav zásob, pri ktorom treba vystaviť objednávku na dodávku, ktorá bude doručená za čas (vopred danej) dodacej lehoty L, takto

$$h = \lambda(L - KC) = \lambda L - KQ, \tag{4.6}$$

kde číslo $K = \lfloor \frac{L}{C} \rfloor$ [3] sa nazýva *počet dodávok na ceste* a číslo KC *zásoby na ceste*.

Pre vypočítanú ekonomickú veľkosť dodávky Q^* potom môžeme z (4.1) získať *optimálnu dĺžku dodávkového cyklu* $C^* = Q^*/\lambda$ a z (4.6) zas *optimálnu hladinu objednania* $h^* = \lambda L - Q^* \cdot \lfloor \frac{\lambda L}{Q^*} \rfloor$.

Ak uvažujeme naviac aj nákupné (obstarávacie) náklady C_N, ktoré sú úmerné objednávanému množstvu Q, dostávame **model s obstarávaním** s nasledujúcou nákladovou funkciou

$$\begin{aligned}\mathcal{H}_1(Q) &= (C_N \cdot Q + C_A + C_S \cdot C \cdot \bar{Q})\frac{1}{C} = C_N\lambda + C_A\frac{\lambda}{Q} + C_S\frac{Q}{2} = \\ &= C_N\lambda + \mathcal{H}(Q),\end{aligned} \tag{4.7}$$

ktorá sa od cieľovej funkcie základného modelu líši len konštantou $C_N\lambda$. Tá nemá vplyv na optimálnu veľkosť dodávky určenej opäť Wilsonovým vzorcom (4.5).

[2] Zhodný vzťah dostaneme, ak položíme ročné akvizičné náklady rovné ročným skladovacím nákladom, t. j. $C_A\frac{\lambda}{Q} = C_S\frac{Q}{2}$. Takýto postup je však korektný len vďaka špeciálnemu tvaru týchto funkcií – všeobecne neplatí.

[3] $\lfloor x \rfloor$ najbližšie menšie celé číslo k x

Variabilné náklady skladovania C_S možno niekedy chápať ako funkciu nákupnej ceny a vyjadriť napríklad v tvare

$$C_S = r \cdot C_N, \tag{4.8}$$

kde r je zlomok $0 < r < 1$, ktorý udáva *ročnú intenzitu výdavkov* na skladovanú jednotku komodity. Wilsonov vzorec (4.5) má potom v tomto **základnom modeli s obstarávaním** tvar

$$Q^* = \sqrt{\frac{2\lambda C_A}{rC_N}} \ . \tag{4.9}$$

V niektorých prípadoch môžeme odhadnúť náklady skladovania C_S pomocou *aktuálnej ceny skladovania* C_K určenej vzťahom

$$C_K = \mathcal{H}_2(Q)/\lambda, \tag{4.10}$$

kde nákladová funkcia $\mathcal{H}_2(Q)$ vznikne z nákladovej funkcie $\mathcal{H}_1(Q)$ nahradením C_S *aktuálnymi ročnými nákladmi* $r \cdot C_K$, t. j.

$$\mathcal{H}_2(Q) = C_N\lambda + C_A\frac{\lambda}{Q} + \frac{QrC_K}{2} \ . \tag{4.11}$$

Potom hovoríme o **modeli s obstarávaním a aktuálnou cenou skladovania**. Zo vzťahov (4.9) a (4.11) dostaneme elementárnymi úpravami

$$C_K = \frac{C_N + \dfrac{C_A}{Q}}{1 - \dfrac{Qr}{2\lambda}}, \tag{4.12}$$

A tak máme po dosadení aktuálnej ceny skladovania do (4.11) nákladovú funkciu v tvare

$$\mathcal{H}_2(Q) = C_N\lambda + C_A\frac{\lambda}{Q} + \frac{r\lambda(C_N \cdot Q + C_A)}{2\lambda - Qr} \ , \tag{4.13}$$

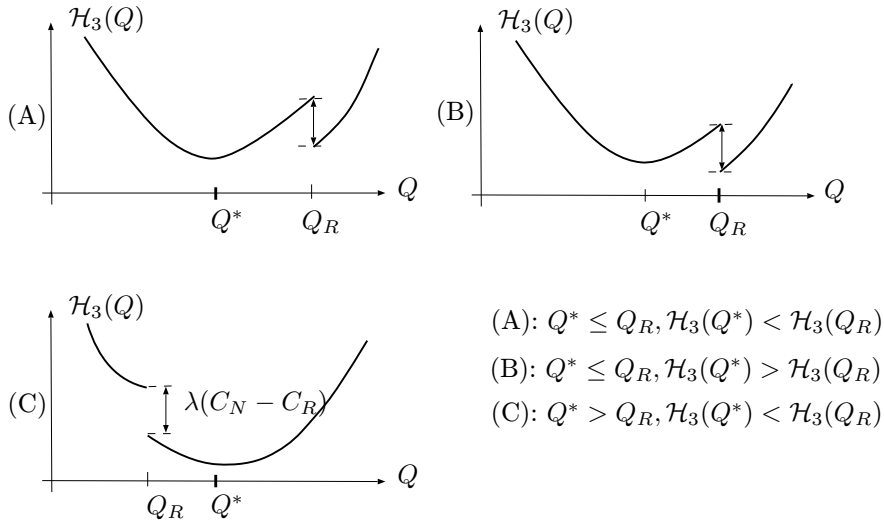
z ktorej ekonomickú veľkosť dodávky Q'^*

$$Q'^* = \frac{2\lambda\phi}{1 + r\phi}, \quad \text{kde } \phi = \sqrt{\frac{C_A}{2\lambda r \cdot C_N + r^2 \cdot C_A}} \tag{4.14}$$

vypočítame opäť pomocou nutnej podmienky (4.4) pre globálne minimum funkcie $\mathcal{H}_2(Q)$.

Možno nájsť príklady parametrov C_N, C_A, λ, r, keď $Q^{'*} < Q^*$, a tak z hľadiska aktuálnych ročných ekonomických nákladov dochádza k ich podhodnoteniu o $\mathcal{H}_2(Q^*) - \mathcal{H}_2(Q^{'*})$ [Sk/rok] voči základnému modelu s obstarávaním.

V súčasnosti je zmysluplný **model s obstarávaním a rabatom**. Tu zostávajú v platnosti predpoklady základného modelu s obstarávaním, ale naviac sa



Obr. 4.3: Prípady nákladovej funkcie $\mathcal{H}_3(Q)$ s obstarávaním a rabatom

predpokladá, že v prípade nákupu väčšieho množstva komodity ako Q_R klesne nákupná cena (v dôsledku poskytnutia rabatu)[4] na $C_R, C_R < C_N$. Nákladová funkcia bude mať v tomto prípade tvar

$$\mathcal{H}_3(Q) = \begin{cases} \lambda C_N + C_A \dfrac{\lambda}{Q} + C_S \dfrac{Q}{2} & \text{ak } Q < Q_R \\ \\ \lambda C_R + C_A \dfrac{\lambda}{Q} + C_S \dfrac{Q}{2} & \text{ak } Q \geq Q_R \end{cases} \tag{4.15}$$

[4]Model možno prirodzeným spôsobom zovšeobecniť na prípad s viacerými cenovými hladinami rabatu.

Vidíme, že nižšia nákupná cena – rabat sa prejaví len v prípade posunu časti nákladovej funkcie $\mathcal{H}_1(Q)$ smerom nadol o hodnotu $\lambda(C_N - C_R)$. Stacionárny bod Q^* je opäť daný Wilsonovým vzorcom (4.5).

Na obr. 4.3 máme možné prípady A,B,C, keď sa dosahujú minimálne náklady buď pri ekonomickej dodávke veľkosti predpovedanej základným modelom Q^* v prípadoch A,C alebo rabatovým množstvom komodity Q_R v prípade B.

Hoci sme sa obmedzili len na spojité verzie Wilsonových modelov – ich *diskrétne* varianty nepredstavujú vážnu komplikáciu. Ukazuje sa totiž, že v prípade analytického hľadania celočíselnej veľkosti dodávok – počtov kusov komodity – často stačí relaxovať príslušný model na spojitý. Diskrétna ekonomická veľkosť dodávky sa určí z minima nákladovej funkcie z nadol a z nahor zaokrúhlených hodnôt zo spojitej veľkosti ekonomickej dodávky. V prípade, ak postačuje len numerický výpočet veľkosti dodávok, výpočtové problémy nevznikajú, nakoľko ich možno realizovať v polynomiálnom čase úplnou enumeráciou.

Doteraz sme modelovali veľkosť spotreby pomocou konštantnej intenzity spotreby λ, čo nám umožnilo vyjadriť veľkosť spotreby za čas t lineárnou funkciou λt. V reálnych úlohách však často pozorujeme nelineárny priebeh spotreby, napr. postupný rast spotreby v počiatočných dodávkových cykloch C_i, ktorý po istom čase τ rastu spotreby prejde do konštantnej spotreby. Ďalej sa obmedzíme len na tento prípad **modelu s rastom spotreby**.

V zhode s prácou [16] budeme predpokladať *mocninový rast* intenzity spotreby v tvare:

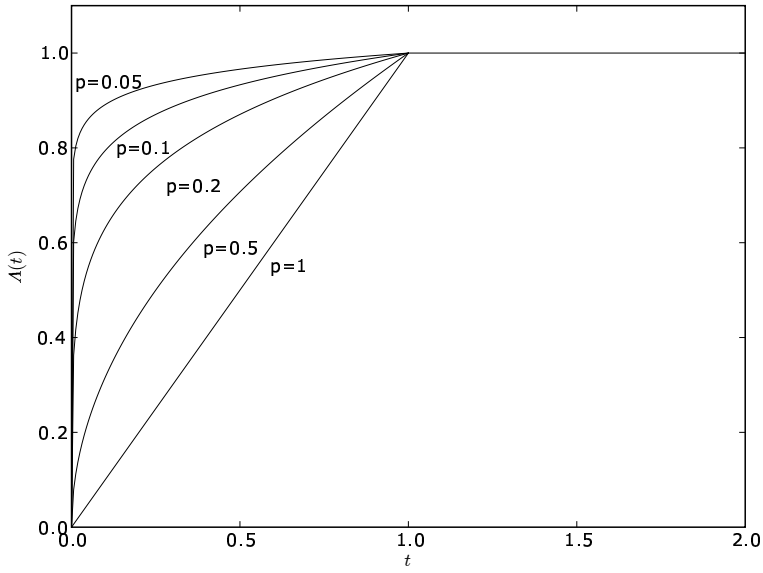
$$\Lambda(t) = \begin{cases} \lambda\left(\frac{t}{\tau}\right)^p & \text{ak } 0 \le t_0 \le t < \tau \\ \lambda & \text{ak } \tau \le t, \end{cases} \tag{4.16}$$

kde parametre $\lambda, \tau, p > 0$, pričom t_0 je čas prvej dodávky na sklad. Tvar funkcie intenzity spotreby $\Lambda(t)$ pre $p < 1$ je na obrázku obr. 4.4, kde je mierka pre čas zvolená tak, aby $\Lambda(1) = 1$. Praktický význam má len prípad $p \le 1$. V prípade $p = 1$ tak máme lineárny rast intenzity spotreby v dobe rastu – nábehu zásobovania τ.

Ďalej sa predpokladá, že je známa optimálna veľkosť dodávky Q^* vypočítaná niektorým z predchádzajúcich Wilsonových modelov. Cieľom je učiť postupnosť dôb dodávok $\{t_i\}_{i=0}^{\infty}$ tak, aby nedochádzalo k deficitu zásob v dodávkových cykloch $C_i = t_{i+1} - t_i$.

Donaldson [10] ukázal, že kumulovaná spotreba v dodávkovom cykle C_i spĺňa nasledujúce rovnice:

$$\int_{t_i}^{t_{i+1}} \Lambda(t)dt = (t_i - t_{i-1})\Lambda(t_i) \quad i = 1, 2, \ldots, n, n+1, \ldots \tag{4.17}$$



Obr. 4.4: Typický priebeh intenzity spotreby
pre rôzne hodnoty p a $\lambda = 1, \tau = 1$

kde t_n je čas poslednej dodávky v čase ukončenia rastu spotreby τ. Po substitúcii $x_i = t_i/\tau$ dostávame rekurentný systém rovníc:

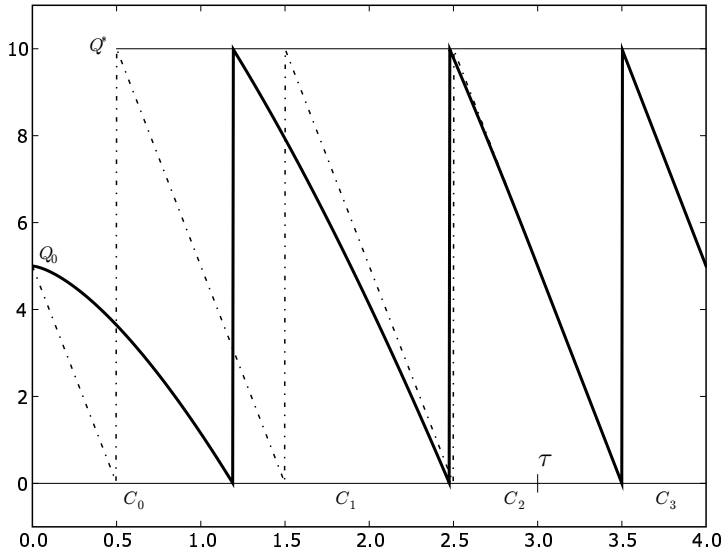
$$\begin{aligned}
(x_{i+1}^{p+1} - x_i^{p+1})/(p+1) &= (x_i - x_{i-1})x_i^p \quad i = 1, 2, \ldots, n-1 \quad n \geq 2 \\
(1 - x_i^{p+1})/(p+1) + (x_{i+1} - 1) &= (x_i - x_{i-1})x_i^p \quad i = n \quad n \geq 1 \\
x_i - x_{i-1} &= Q^*/\lambda\tau \quad i = n+1, \ldots \quad n \geq 1 \\
x_1 - x_0 &= Q^*/\lambda\tau \quad n = 0
\end{aligned}$$

ktorý umožňuje priamo vypočítať pre dané x_1 a x_0 postupnosť x_i až x_{n+1}. Možno overiť, že dĺžka dodávkových cyklov postupne klesá, kým nedosiahne hodnotu $C = Q^*/\lambda = \tau(x_{n+1} - x_n)$. V krajnom prípade, keď $x_1 > 1$, je potom $n = 0$ a $Q^* = \lambda\tau(x_1 - x_0)$.

Ostáva ešte vypočítať veľkosť počiatočnej dodávky Q_0 v dodávkovom cykle C_0 z danej hodnoty x_1 zo vzťahu

$$Q_0/\lambda\tau = \int_{t_0}^{t_1} \Lambda(t)dt/\lambda\tau = \begin{cases} (x_1^{p+1} - x_0^{p+1})/(p+1) & \text{ak } x_1 < 1 \\ x_1 - (x_0^{p+1} + p)/(p+1) & \text{ak } x_1 \geq 1. \end{cases}$$

Principiálne je teda možné pre dané t_0 a p tabelovať hodnoty $t_1 : t_0 < t_1 \leq Q^*/\lambda$ a vypočítať príslušné Q_0 a prípadne ich zobraziť graficky.



Obr. 4.5: Vývoj stavu zásob
v modeli s mocninovým rastom spotreby ($p = 0.5$)

Na obrázku obr. 4.5 je hrubou čiarou zobrazený vývoj stavu zásob pri dodávke $Q_0 = 5$ v počiatočnom dodávkovom cykle C_0 a následných optimálnych dodávkach veľkosti $Q^* = 10$ v postupne sa skracujúcich cykloch C_1 a C_2. Doba rastu spotreby je ukončená v čase $\tau = 3$, a tak ďalšie cykly $(C_3, C_4 \ldots)$ po dobe rastu sú s konštantnou intenzitou spotreby $\lambda = 10$ a majú konštantnú dĺžku $C^* = Q^*/\lambda$. Bodkočiarkované čiary zodpovedajú vývoju stavu zásob pri

konštantnej spotrebe. Ak teda rešpektujeme mocninový rast spotreby, ušetríme jeden dodávkový cyklus.

Uvedený prístup môžeme aplikovať aj v **modeli s útlmom spotreby**, kde je známy čas, po ktorom sa očakáva, že spotreba komodity postupne začne klesať, až klesne na nulu. Tu sa však predpokladá konečný horizont – interval $\langle t_Z, t_K)$ skladovania komodity.

Spojením týchto prístupov môžeme analogicky odvodiť **model s rastom a útlmom spotreby**. Cieľom je opäť nájsť časy dodávok v období rastu spotreby $\langle t_Z, \tau_R)$, v období ustálenej (konštantnej) spotreby $\langle \tau_R, t_K - \tau_U)$ a v období útlmu spotreby $\langle t_K - \tau_U, t_K)$. Časy τ_R a τ_U tu označujú koniec doby rastu a začiatok doby útlmu spotreby.

4.2.2 Modely s deficitom

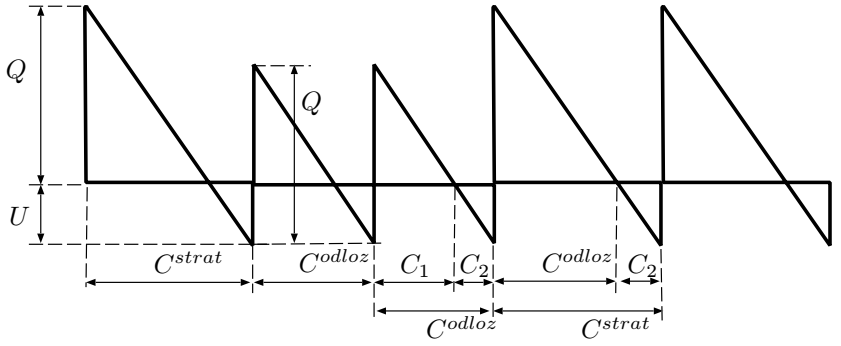
Budeme sa zaoberať reálnymi situáciami, keď vznikne požiadavka na výber komodity zo skladu, ale ju nemôžeme uspokojiť, pretože táto komodita v sklade chýba. Podľa spôsobu riešenia problému *deficitu zásob* sa rozlišujú modely *so stratenou spotrebou* a modely *s odloženou spotrebou*.

V **modeli s alternatívnou spotrebou** predpokladáme, že neuspokojená požiadavka na spotrebu bude s pravdepodobnosťou ψ pokrytá z nasledujúcej dodávky a s pravdepodobnosťou $1 - \psi$ ostane nepokrytá. Pre $\psi = 1$ dostávame **model s odloženou spotrebou** a pre $\psi = 0$ **model so stratenou spotrebou**.

S deficitom zásob sú spojené *ročné (jednotkové) náklady deficitu* C_D^{strat} pri stratených predajoch a C_D^{odloz} pri odloženej spotrebe, ktoré môžeme chápať ako dva druhy penalizácie 1 jednotky komodity za rok. Výšku deficitu, t. j. počet neuspokojených požiadaviek v jednom dodávkovom cykle, budeme značiť U. Predpokladajme pre jednoduchosť, že ostatné predpoklady základného Wilsonovho modelu ostávajú v platnosti.

Príklad vývoja stavu zásob v modeli s alternatívnou spotrebou je znázornený na obr. 4.6, kde na začiatku dodávkových cyklov C^{odloz} sa deficit zásob rieši odloženou spotrebou a na začiatku dodávkových cyklov C^{strat} stratenou spotrebou.

Dodávkový cyklus pri odloženej spotrebe $C^{odloz} = C_1 + C_2$ sa skladá z dvoch období: C_1, keď sú uspokojené všetky požiadavky na výber komodity zo skladu, a C_2, keď vzniká deficit. Kumulovaná výška zásob v období C_1 je $(Q - U)C_1/2$ a platí $(Q - U)/Q = C_1/C^{odloz}$, a tak je priemerný stav zásob v dodávkovom



Obr. 4.6: Vývoj stavu zásob v modeli s alternatívnou spotrebou

cykle C^{odloz}

$$\bar{Q}^{odloz} = \frac{1}{C^{odloz}} \cdot \frac{(Q-U)C_1}{2} = \frac{(Q-U)^2}{2Q} \ . \tag{4.18}$$

Dodávkový cyklus pri stratených predajoch $C^{strat} = C^{odloz} + C_2$ sa skladá z dvoch období: C^{odloz}, keď sú uspokojené všetky požiadavky na výber komodity zo skladu, a C_2, keď vzniká deficit. Kumulovaná výška zásob v období C^{odloz} je $QC^{odloz}/2$ a platí $(Q+U)/Q = C^{strat}/C_2$, a tak je priemerný stav zásob v dodávkovom cykle C^{strat}

$$\bar{Q}^{strat} = \frac{1}{C^{strat}} \cdot \frac{Q \cdot C^{odloz}}{2} = \frac{Q^2}{2(Q+U)} \ . \tag{4.19}$$

Kumulovaná výška deficitu je vzhľadom na konštantnú intenzitu spotreby v oboch dodávkových cykloch $U \cdot C_2/2$, a tak neprekvapuje, že priemerný deficit v dodávkovom cykle C^{odloz} aj C^{strat} bude rovný tej istej hodnote

$$\bar{U} = \frac{U^2}{2Q} \ . \tag{4.20}$$

Priemerná ročná nákladová funkcia $\mathcal{H}_4(Q,U)$ je potom tvorená priemernými ročnými nákladmi za dodávku, skladovanie a deficit zásob

$$\begin{aligned} \mathcal{H}_4(Q,U) \ &= (C_A + C_S \cdot C^{odloz} \cdot \bar{Q}^{odloz} + C_D^{odloz} \cdot \bar{U}) \cdot \frac{\psi}{C^{odloz}} + \\ &+ \ (C_A + C_S \cdot C^{strat} \cdot \bar{Q}^{strat} + C_D^{strat} \cdot \bar{U}) \cdot \frac{1-\psi}{C^{strat}} \ . \end{aligned}$$

Po elemetárnych úpravách s využitím vzťahov $\lambda = Q/C^{odloz} = (Q + U)/C^{strat}$ dostaneme

$$\begin{aligned}\mathcal{H}_4(Q,U) =& C_A\lambda\Big(\frac{\psi}{Q} + \frac{1-\psi}{Q+U}\Big) + C_S\Big(\frac{\psi(Q-U)^2}{2Q} + \frac{(1-\psi)Q^2}{2(Q+U)}\Big) \\ &+ C_D^{odloz}\frac{\psi\lambda U^2}{2Q^2} + C_D^{strat}\frac{(1-\psi)\lambda U^2}{2Q(Q+U)}\ . \end{aligned} \tag{4.21}$$

Optimalizačná úloha vedie na hľadanie minima nákladovej funkcie (4.18) s nutnou podmienkou

$$\begin{aligned}\frac{\partial\mathcal{H}_4(Q,U)}{\partial Q} =& - C_A\lambda\Big(\frac{\psi}{Q^2} + \frac{1-\psi}{(Q+U)^2}\Big) + C_S\Big(\frac{\psi(Q^2-U^2)}{2Q^2} + \frac{(1-\psi)Q(Q+2U)}{2(Q+U)^2}\Big) + \\ &- C_D^{odloz}\frac{\psi\lambda U^2}{Q^3} - C_D^{strat}\frac{(1-\psi)\lambda U^2(2Q+U)}{Q^2(Q+U)} = 0\ , \end{aligned} \tag{4.22}$$

$$\begin{aligned}\frac{\partial\mathcal{H}_4(Q,U)}{\partial U} =& - C_A\lambda\frac{1-\psi}{(Q+U)^2} - C_S\Big(\frac{2\psi(Q-U)}{Q} + \frac{(1-\psi)Q^2}{2(Q+U)^2}\Big) + \\ &+ C_D^{odloz}\frac{\psi\lambda U}{Q^2} + C_D^{strat}\frac{(1-\psi)\lambda U(U-2Q)}{Q(Q+U)^2} = 0\ , \end{aligned} \tag{4.23}$$

v tvare systému dvoch nelineárnych rovníc (4.22) a (4.23), ktorý už neumožňuje vyjadriť riešenia Q^*, U^* explicitne – v tvare nezávislých vzorcov. Pri ich numerickom hľadaní najčastejšie vhodnými iteračnými metódami[5] sa stretávame s problémom počiatočného riešenia.

Počítačové experimenty ukazujú, že v mnohých prípadoch stačí vyjsť z počiatočného riešenia určeného Wilsonovým vzorcom (4.5) a hľadať Q^* s presnosťou $\varepsilon > 0$. Nevýhodou však je, že nie je zaručená konvergencia metódy – môže sa aj zacykliť (čo ale ľahko odstránime dodatočným kritériom, napr. obmedzením počtu iterácií). Príkladom takého hľadania riešenia je nasledujúca iteračná metóda:

Algoritmus 4.1. Iteračná metóda

Krok 1: Položíme $Q_1 = \sqrt{\dfrac{2\lambda C_A}{C_S}}$.

[5]Malé systémy nelineárnych rovníc možno úspešne riešiť aj v prostredí známych tabuľkových procesorov, napr. EXCEL pod Windows, resp. GNUMERIC pod Linux.

Krok 2: Po dosadení Q_1 do ľavej strany rovnice (4.22) dostaneme nelineárnu rovnicu $f(U) = 0$ premennej U a nech je jej riešením U_1.

Krok 3: Po dosadení U_1 do ľavej strany rovnice (4.23) dostaneme nelineárnu rovnicu $g(Q) = 0$ premennej Q a nech je jej riešením Q_2.

Krok 4: Ak $|Q_1 - Q_2| < \varepsilon$, potom *STOP* s riešením $Q^* = Q_1, U^* = U_1$. Ináč položíme $Q_1 = Q_2$ a **GOTO Krok 2**.

V prípade, že nie je vopred známa relatívna početnosť pokrývania deficitu zásob odhadujúca pravdepodobnosť ψ (voľby odloženej spotreby), môžeme ju považovať za ďalšiu premennú. Potom priemerná ročná nákladová funkcia $\mathcal{H}_5(Q, U, \psi)$ bude opäť tvare (4.21), ale s trojicou premenných. Jej minimalizáciou dostávame okrem optimálnej veľkosti dodávky Q^* a optimálnej veľkosti deficitu U^* aj stratégu (politiku) ψ^*, s akou sa treba rozhodovať pre odloženú spotrebu voči alterntívnym strateným predajom.

Premennú U môžeme interpretovať aj ako výšku *poistnej zásoby*, z ktorej čerpáme s pravdepodobnosťou ψ v čase deficitu bežných zásob v sklade. Optimálna hodnota U^* nám umožňuje rozhodnúť o ekonomickej veľkosti poistnej zásoby, ktorá je v tomto modeli obnovená jednorazovo z dodávky na začiatku následného dodávkového cyklu.

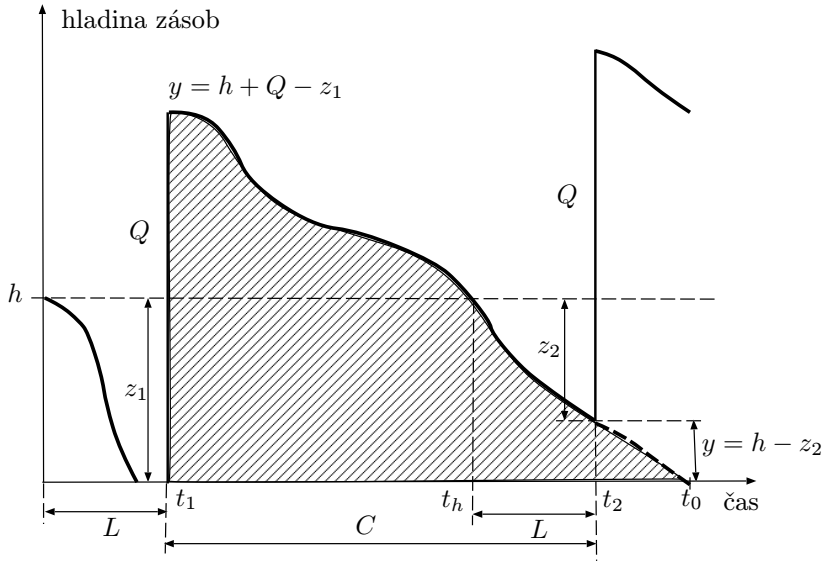
Ďalšou interpretačnou možnosťou je považovať ψ za pravdepodobnosť príchodu dodávky veľkosti $Q + U$ voči alternatívnej dodávke veľkosti Q, ktoré vedú pri konštantnej (deterministickej) spotrebe k dodávkovým cyklom C^{odloz} a C^{strat}.

Alternatívny model spotreby je príkladom veľmi jednoduchého stochastického modelu zásob, v ktorom sa náhodnosť týka len voľby riešenia deficitu zásob ale spotreba je modelovaná deterministicky (lineárnou funkciou).

V **stochastickom** (h, Q) **modeli so stratenou spotrebou** je pozornosť sústredená na určenie hladiny objednania h a jednorazovej dodávky veľkosti $Q, Q > h \geq 0$ pri náhodnej spotrebe známej intenzity λ.

Pre jednoduchosť budeme predpokladať, že dodacia lehota L je vždy menšia než dodávkový cyklus $C, (L < C)$, pričom L je neznáma konštanta a C v dôsledku náhodnej spotreby náhodná veličina.

Dynamika stavu zásob v dodávkovom cykle je zobrazená na obr. 4.7. Dodávkový cyklus C začína v čase t_1 po príchode dodávky veľkosti Q a končí v čase t_2 po príchode ďalšej dodávky veľkosti Q. V čase t_h, keď hladina zásob klesne na úroveň h, je podaná objednávka, ktorá je realizovaná za dobu L. V čase t_1 je stav zásob $y = h + Q - z_1$ a v čase t_2 (na konci cyklu) je stav zásob $y = h - z_2$, pričom z_1, z_2 sú *zostatky zásob* v príslušných dodacích lehotách.



Obr. 4.7: Stav zásob v modeli (h,Q) so stratenou spotrebou

Nech je náhodná spotreba počas dodacej lehoty L určená distribučnou funkciou $F(x), x \geq 0$. Potom *priemerný deficit zásob* $\bar{U}(h)$ (v dodacej lehote) je

$$\bar{U}(h) = \begin{cases} \lambda L & \text{ak } h = 0 \\ \int_h^\infty (z-h)dF(z) & \text{ak } h > 0 \end{cases} \tag{4.24}$$

funkciou hladiny objednania h. *Priemerná dĺžka dodávkového cyklu* $\bar{C}(h, Q)$ je potom rovná

$$\bar{C}(h, Q) = \frac{Q + \bar{U}(h)}{\lambda}. \tag{4.25}$$

Distribučnú funkciu *zostatku zásob* $G(z)$ v dodacej lehote definujeme pomocou distribučnej funkcie $F(z)$ spotreby takto

$$G(z) = \begin{cases} F(z) & \text{ak } z \leq h \\ 1 & \text{ak } z > h \end{cases} \tag{4.26}$$

Priemerný zostatok zásob $\bar{Z}(h)$ vypočítame pomocou vzťahu (4.24)

$$\bar{Z}(h) = \int_0^\infty z\ dG(z) = \int_0^h z\ dF(z) + h\int_h^\infty dF(z)$$
$$= \int_0^\infty z\ dF(z) - \int_h^\infty (z-h)dF(z) = \lambda L - \bar{U}(h). \qquad (4.27)$$

Na obr. 4.7 vidíme, že v čase t_0 sú „staré“ zásoby vyčerpané, a tak vyšrafovaná plocha zodpovedá *starým kumulovaným zásobám* $S(y)$. Priemernú spotrebu predpokladáme spojitú, stacionárnu a nezápornú, a tak priemerné zásoby v intervale $\langle t_1, t_0\rangle$ sú $y/2$, pričom priemerná dĺžka intervalu $\langle t_1, t_0\rangle$ je y/λ, takže

$$S(y) = \frac{y^2}{2\lambda}.$$

Priemerné kumulované zásoby $\bar{Q}_K(h,Q)$ v dodávkovom cykle potom dostaneme zo vzťahu

$$\bar{Q}_K(h,Q) = \int_0^\infty S(Q+r-z)dG(z) - \int_0^\infty S(r-z)dG(z)$$
$$= \frac{Q}{2\lambda}\int_0^\infty (Q-2h-2z)dG(z) = \frac{Q}{\lambda}\cdot\Big(\frac{Q}{2}+h-\bar{Z}(h)\Big). \qquad (4.28)$$

Priemerná ročná nákladová funkcia $\mathcal{H}_6(h,Q)$ je opäť tvorená priemernými ročnými nákladmi za dodávku, skladovanie a deficit zásob

$$\mathcal{H}_6(h,Q) = (C_A + C_S\cdot\bar{Q}_K(h,Q) + C_D\cdot\bar{U}(h))/\bar{C}(h,Q). \qquad (4.29)$$

Výpočet ekonomickej veľkosti dodávky Q^* a optimálnej hladiny objednania h^* je opäť možné numericky minimalizáciou nákladovej funkcie (4.29).

V práci [46] sú odvodené implicitné formuly pre prípady logaritmicko konkávnych[6] distribučných funkcií $F(x)$. Najčastejšie ide o gamma, Weibullovo, logistické, normálne a logaritmicko-normálne rozdelenia aj ich odseknuté verzie.

4.3 Dynamické modely zásob

V predchádzajúcich modeloch (s výnimkou modelov rastu, resp. útlmu spotreby) sme predpokladali stacionárny priebeh spotreby (napr. konštantná spotreba,

[6]Reálnu funkciu $f : \mathcal{D} \to \mathbb{R}$ nazývame logaritmicko-konkávnou, ak je konkávny jej prirodzený logaritmus, t. j. ak platí $f(\alpha x_1 + (1-\alpha)x_2) \geq f(x_1)^\alpha f(x_2)^{1-\alpha}$ pre ľubovoľné $x_1, x_2 \in \mathcal{D}$ a ľubovoľné $\alpha \in \langle 0,1\rangle$.

známe pravdepodobnostné rozdelenie spotreby) v dodávkových cykloch. V dynamických modeloch sa predpokladá, že v plánovanom horizonte zásobovania $\mathcal{T} = \{1, 2, \dots\}$ sú v periódach t známe očakávané veľkosti spotreby $S_t, t \in \mathcal{T}$. Ďalej sa obmedzíme na modely s konečným horizontom – konečným počtom periód $T = |\mathcal{T}|$ a tak $\mathcal{T} = \{1, 2, \dots, T\}$.

V **modeloch s jednou komoditou** treba určiť aké množstvo x_t tejto komodity vyrobiť pri maximálnej produkcii (kapacite výroby) K_t jednotiek komodity, ak sa má pokryť celá spotreba každej periódy t. Tento klasický **Wagnerov a Whitinov model** nepripúšťa možnosť vzniku deficitu zásob, ktorému sa vyhýba presunom zásob neznámej veľkosti z_t z periódy t do periódy $t + 1$. Sú známe jednotkové náklady na výrobu komodity C_t a náklady na skladovanie jednotky komodity H_t z periódy t do periódy $t + 1$.

Takto formulovaný problém možno riešiť ako nasledujúcu úlohu lineárneho programovania (WW):

$$\min \sum_{t \in \mathcal{T}} C_t x_t + H_t z_t \tag{4.30}$$

$$\text{za predpokladov:} \quad z_{t-1} + x_t - z_t = S_t \qquad t \in \mathcal{T} \tag{4.31}$$

$$0 \le x_t \le K_t \qquad t \in \mathcal{T} \tag{4.32}$$

$$z_0 = 0, z_t \ge 0 \qquad t \in \mathcal{T} \tag{4.33}$$

Cieľová funkcia (4.30) udáva celkové náklady spojené s výrobou a skladovaním komodity, ktoré treba minimalizovať. Podmienka (4.31) zabezpečuje pokrytie celej spotreby a podmienka (4.32) obmedzuje produkciu výroby maximálnou kapacitou v každej perióde z $\mathcal{T}$. Nulová hodnota pomocnej premennej z_0 sa interpretuje ako nulové zásoby komodity na začiatku prvej periódy.

Úlohu WW môžeme efektívne riešiť simplexovou metódou na riešenie úloh lineárneho programovania (LP). Možno ukázať, že v prípade celočíselných spotrieb S_t a kapacít K_t je i riešenie úlohy celočíselné. Ak totiž z rovníc (4.31) vyjadríme premenné $x_t = S_t + z_t - z_{t-1}$ a dosadíme do rovníc (4.32) dostávame nasledujúcu redukovanú úlohu (RWW):

$$\min \sum_{t \in \mathcal{T}} H_t^R z_t \tag{4.34}$$

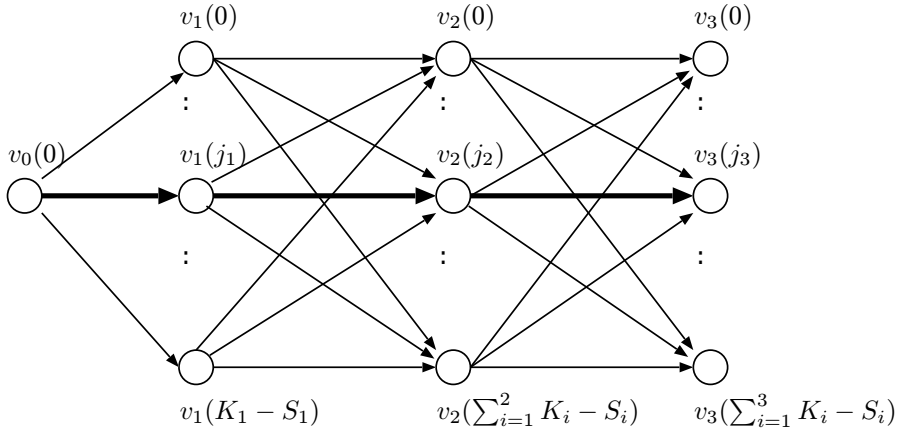
$$\text{za predpokladov:} \quad -S_t \le z_t - z_{t-1} \le K_t - S_t \qquad t \in \mathcal{T} \tag{4.35}$$

$$z_0 = 0, z_t \ge 0 \qquad t \in \mathcal{T} \tag{4.36}$$

kde

$$H_t^R = \begin{cases} C_t - C_{t+1} + H_t & \text{ak } 1 \le t \le T-1 \\ C_t + H_t & \text{ak } t = T \end{cases} \tag{4.37}$$

Úlohu RWW môžeme riešiť aj ako úlohu o hľadaní najkratšej cesty v hranovo ohodnotenom acyklickom digrafe $\mathcal{G} = (\mathcal{V}, \mathcal{H}, \gamma)$.



Obr. 4.8: Cesta $v_0(0) \to v_1(j_1) \to v_2(j_2) \to v_3(j_3)$
v acyklickom digrafe $\mathcal{G} = (\mathcal{V}, \mathcal{H}, \gamma)$
je priradená riešeniu $z_0 = 0, z_1 = j_1, z_2 = j_2, z_3 = j_3$ úlohy RWW

Na obrázku obr. 4.8 máme príklad takého acyklického digrafu priradenému úlohe RWW s periódami $\mathcal{T} = \{1, 2, 3\}$ Množina vrcholov $\mathcal{V}$ je tvorená vrcholmi $v_t(k)$ zodpovedajúcimi hodnote premennej $z_t = k$ pričom vrcholom $v_0(0)$ je počiatočný vrchol, formálne

$$\mathcal{V} = \big\{v_0(0)\big\} \cup \big\{v_t(j) \mid t \in \mathcal{T},\ j \in \{0, 1, \ldots, \textstyle\sum_{i=1}^{t} K_i - S_i\}\big\}.$$

Množina hrán $\mathcal{H}$ spája vrcholy digrafu, ktoré vyhovujú podmienkam (4.35) a (4.36), t. j.

$$\mathcal{H} = \big\{(v_{t-1}(a), v_t(b)) \in \mathcal{V} \times \mathcal{V} \mid\ -S_t \le b - a \le K_t - S_t,\ t \in \mathcal{T}\big\}.$$

Ohodnotenie hrán $\gamma : \mathcal{H} \to \mathbb{R}$ sa potom definuje v zhode s cieľovou funkciou (4.34) nasledujúco:

$$\gamma\big(v_{t-1}(a), v_t(b)\big) = H_t^R \cdot b.$$

Cieľom je nájsť najkratšiu $v_0(0) \to v_1(j_1) \to \cdots \to v_T(j_T)$ cestu, ktorej zodpovedá riešenie $z_0 = 0, z_1 = j_1, \ldots z_T = j_t$. Príslušné hodnoty x_1, x_2, x_3 produkcie výroby úlohy WW môžeme vypočítať z rovníc (4.31).

Úlohy WW a RWW však nemusia mať vždy prípustné riešenie z dôvodu obmedzenej kapacity výroby, keď kumulovaná spotreba v niektorej perióde $r \in \mathcal{T}$ prevyšuje kumulovanú produkciu výroby $\sum_{k=1}^{r}(x_k - S_k) < 0$. Iným dôvodom môže byť aj niekedy naviac uvažovaná kapacita skladu M_t udávajúca maximálne množstvo skladovanej komodity v perióde t. Riešením môže byť opäť pripustenie *deficitu zásob* v niektorých periódach, ktoré sa rieši – ako sme ukázali – buď odloženou spotrebou ($\delta = 1$) alebo stratenou spotrebou ($\delta = 0$).

Nech naviac premenné y_t udávajú veľkosť deficitu v perióde t a sú ocenené nákladmi D_t za jednotku chýbajúcej komodity. V prípade odloženej spotreby ide o dodatočné náklady spojené s odkladom spotreby, zatiaľčo pri stratených predajoch ide o ušlý zisk z možného predaja. Dostávame **Wagnerov a Whitinov model s deficitom**, ktorý môžeme riešiť ako nasledujúcu úlohu lineárneho programovania (WWD):

$$\min \sum_{t \in \mathcal{T}} C_t x_t + H_t z_t + D_t y_t \tag{4.38}$$

$$\text{za predpokladov:} \quad z_{t-1} - \delta y_{t-1} + x_t - z_t + y_t = S_t \qquad t \in \mathcal{T} \tag{4.39}$$

$$0 \leq x_t \leq K_t \qquad t \in \mathcal{T} \tag{4.40}$$

$$z_0 = 0, 0 \leq z_t \leq M_t \qquad t \in \mathcal{T} \tag{4.41}$$

$$y_0 = 0, y_t \geq 0 \qquad t \in \mathcal{T} \tag{4.42}$$

Cieľová funkcia (4.38) udáva celkové náklady spojené s produkciou, tvorbou zásob v sklade a nákladmi za deficit zásob. Podmienka (4.39) rieši bilanciu medzi zásobami a spotrebou v perióde t. Pri odloženej spotrebe ($\delta = 1$) zabezpečuje pokrytie odloženej spotreby z predchádzajúcej periódy y_{t-1}, ako aj možnosť vytvorenia ďalšej odloženej spotreby y_t. Pri stratených predajoch ($\delta = 0$) zabezpečuje vytvorenie nerealizovanej spotreby veľkosti y_t. Podmienky (4.40) a (4.41) rešpektujú obmedzenú kapacitu výroby a skladu. Podmienka (4.42) je obligatórnou pre veľkosť deficitu. Nulové hodnoty pomocných premenných z_0 a y_0 znamenajú, že pred prvou periódou nemáme žiadne zásoby v sklade ani žiaden dlh v podobe deficitu zásob. Opäť možno dokázať existenciu celočíselného riešenia pri celočíselných obmedzeniach úlohy pomocou simplexovej metódy pre úlohy LP.

Iný praktický **Wagnerov a Whitinov model s prerušením výroby** riešil Hindi [17], keď navrhol penalizovať kladnými hodnotami aj začatie výroby nákladmi P_t po každom prerušení výroby, aj samotný chod výroby nákladmi R_t.

Označme u_t bivalentnú premennú rovnú 1, ak sa uvažovaná komodita vyrába a 0, ak sa nevyrába. Ďalšia bivalentná premenná w_t je rovná 1, len ak je začatá

výroba v perióde t. Dostávame tak úlohu zmiešaného programovania (WWP):

$$\min \sum_{t \in \mathcal{T}} C_t x_t + H_t z_t + P_t u_t + R_t w_t \tag{4.43}$$

$$\text{za predpokladov:} \quad z_{t-1} + x_t - z_t = S_t \quad t \in \mathcal{T} \tag{4.44}$$

$$x_t - K_t u_t \leq 0 \quad t \in \mathcal{T} \tag{4.45}$$

$$z_0 = 0, z_t \geq 0 \quad t \in \mathcal{T} \tag{4.46}$$

$$w_t - u_t + u_{t-1} \leq 0 \quad t \in \mathcal{T} \tag{4.47}$$

$$x_t \geq 0 \quad t \in \mathcal{T} \tag{4.48}$$

$$u_t \in \{0,1\}, u_0 = 0 \quad t \in \mathcal{T} \tag{4.49}$$

$$w_t \in \{0,1\} \quad t \in \mathcal{T} \tag{4.50}$$

$$\tag{4.51}$$

Cieľová funkcia (4.43) opäť udáva celkové náklady spojené s výrobou a skladovaním komodity, ktoré treba minimalizovať. Podmienky (4.31) a (4.33) sú zhodné s podmienkami (4.44) a (4.46). Výroba produkuje nanajvýš K_t jednotiek len ak je v chode, čo vedie k podmienke (4.45). Predpokladá sa, že pred prvou periódou sa nevyrába, čo vedie k nulovej hodnote pomocnej premennej u_0. Skutočnosť, že $R_t > 0$, zabezpečuje, že $w_t = 1$ len vtedy, keď $u_t = 1$ a $u_{t-1} = 1$. Na riešenie takto formulovanej úlohy autor uvádza metódu vetiev a hraníc, ktorá sa ukázala pomerne efektívnou pre úlohy malých rozmerov $|\mathcal{T}| \leq 12$.

Úlohu WWP možno doplniť o požiadavky úlohy WWD, čo vedie k úlohe **Wagnerov a Whitinov model s prerušením výroby a deficitom zásob**, ktorú by bolo možné pre úlohy malých rozmerov riešiť metódou vetiev a hraníc. Ďalšie zovšeobecnenia sa týkajú rôznych *viackomoditných verzií Wagner-Whitinových modelov*, ktoré v najjednoduchšom prípade nahradzujú silné separované obmedzenia kapacít skladu, resp. výroby komodít k ich slabším kumulovaným ohraničeniam.

V prípade heuristického riešenia prichádzajú do úvahy rôzne aproximatívne verzie metódy vetiev a hraníc, ktoré síce nezaručujú optimalitu riešenia, poskytujú riešenie v prijateľnom čase.

4.4 Frontové modely zásob

Problémy ekonomickej *kapacity skladov* môžeme modelovať pomocou elemetárnych systémov hromadnej obsluhy [18]. Podľa interpretácie vstupného toku zákazníkov rozlišujeme modely so

- *vstupným tokom dodávok*, keď tok zákazníkov reprezentuje jednotky dodávky a operácia obsluhy zákazníka predstavuje uspokojenie jednotky spotreby,

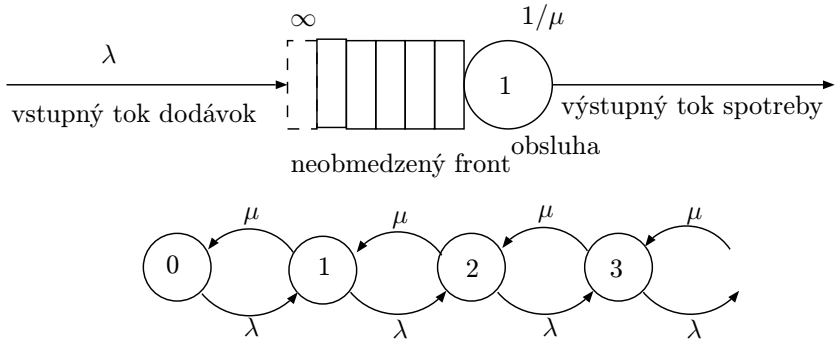
- *vstupným tokom spotrieb*, keď tok zákazníkov je tok jednotiek spotreby komodity a obsluha predstavuje dopĺňanie zásob v čase dodacej lehoty.

V elemetárnych markovovských modeloch zásob, na ktoré sa ďalej sústredíme, je vstupný tok zákazníkov modelovaný Poissonovým procesom a doba obsluhy zákazníka je exponenciálna.

Modely zásobovacích reťazcov sa môžu riešiť ako markovovské siete. Uzlami sú elementárne systémy zásob a hranami príslušné vstupné/výstupné (medziskladové) väzby. Príslušný matematický aparát umožňuje rozhodovať aj aktuálne otázky týkajúce sa ekonomických dôsledkov *výluk a umiestnenia skladov*.

4.4.1 Modely so vstupným tokom dodávok

Najskôr uvažujme jednoduchý **frontový model** $M/M/1/\infty$ [7] **so vstupným tokom dodávok**, schéma a prechodový graf je na obr. 4.9.



Obr. 4.9: Schéma a prechodový graf systému $M/M/1/\infty$

Vstupný tok dodávok je modelovaný Poissonovým procesom s intenzitou λ, a tak dĺžka medzery medzi príchodmi jednotkových dodávok do skladu má

[7] V Kendallovej klasifikácii elementárnych systémov hromadnej obsluhy $A/B/n/m$ charakterizujú písmená A – vstupný tok zákazníkov a B - dobu obsluhy zákazníka jednou linkou obsluhy, n – počet nezávislých liniek obsluhy a m – maximálny počet zákazníkov v systéme.

strednú hodnotu $\frac{1}{\lambda}$. Doba obsluhy dodávky (jedinou obslužnou linkou realizujúcou vyskladnenie dodávky) zodpovedá dobe uskladnenia a má exponenciálne rozdelenie s parametrom μ. Jednotka zásob je tu teda rovná jednotke dodávky a je po strednej dobe $\frac{1}{\mu}$ vyskladnená.

Množina stavov systému $S = \{0, 1, 2, \dots\}$ predstavuje počet jednotiek zásob v sklade. Stav 0 reprezentuje stav deficitu zásob, keď nemôže byť žiadna jednotka spotreby uspokojená. Systém je zaujímavý po (časovej) stabilizácii, t. j. v prípade $\rho = \frac{\lambda}{\mu} < 1$.

Z prechodového grafu možno ľahko odvodiť stacionárne rozdelenia stavov systému $(\pi_0, \pi_1, \dots), \pi_j = \rho^j(1-\rho)$. *Priemernú výšku zásob v sklade* odhadneme stredným počtu zákazníkov v systéme (započítava sa aj vyskladňovaná zásoba)

$$\bar{Q} = \sum_{j=0}^{\infty} j\pi_j = \frac{\rho}{1-\rho} \ . \tag{4.52}$$

Optimalizácia sa aj v modeloch tohto typu realizuje hľadaním minima nákladovej funkcie $\mathcal{H}_7(\rho)$ zloženej z priemerných jednotkových nákladov skladovania a z priemerných jednotkových nákladov deficitu (pri známych porovnateľných nákladoch C_S a $C_D\lambda$, $C_S < C_D\lambda$)

$$\mathcal{H}_7(\rho) = C_S \cdot \bar{Q} + C_D \cdot \lambda\pi_0 = C_S\frac{\rho}{1-\rho} + C_D\lambda(1-\rho), \tag{4.53}$$

s optimálnym riešením

$$\rho^* = 1 - \sqrt{\frac{C_S}{\lambda C_D}} \ .$$

Dostávame naviac aj odpoveď na otázku, aký by mal byť optimálny pomer $\rho^* = \frac{1/\mu}{1/\lambda}$ medzi priemernou dobou medzi príchodmi dodávky do skladu $1/\lambda$ a priemernou dobou medzi vyskladneniami (spotrebami) dodávok $1/\mu$.

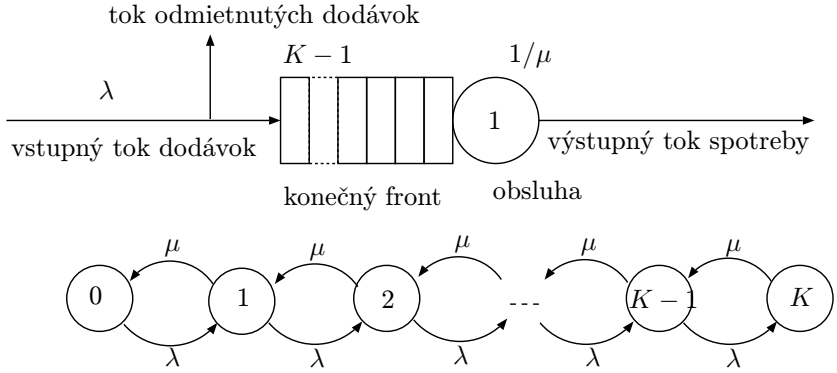
Nevýhodou optimalizácie nákladovej funkcie $\mathcal{H}_7(\rho)$ ako voľného extrému je, že neoptimalizuje priemerný stav zásob, ktorý tak môže nadobudnúť neprijateľné hodnoty. Tento nedostatok čiastočne odstraňuje obmedzenie P_q pravdepodobnosti, že stav zásob prekročí určitú kritickú hodnotu q

$$\mathcal{P}(Q > q) = \sum_{j=q+1}^{\infty} \pi_j = \rho^{q+1} \leq P_q \ .$$

čo už vedie k optimalizačnej úlohe o viazanom extréme s riešením ρ^{**}

$$\mathcal{H}_7(\rho^{**}) = \min\{\mathcal{H}_7(\rho) \mid 0 < \rho^{q+1} \leq P_q\}. \tag{4.54}$$

Inou možnosťou je voľba konečného (nenulového) **frontového modelu** $M/M/1/K$ **so vstupným tokom dodávok**, so schémou a prechodovým grafom na obr. 4.10.



Obr. 4.10: Schéma a prechodový graf systému $M/M/1/K$

Parameter K, $K > 1$ interpretujeme ako (konečnú) kapacitu skladu, ostatné parametre tu majú význam ako v predchádzajúcom modeli. V dôsledku konečnej (obmedzenej) kapacity skladu však môžu byť niektoré z prichádzajúcich dodávok odmietnuté.

Množina stavov systému $S = \{0, 1, 2, \ldots, K\}$ predstavuje opäť počet jednotiek zásob v sklade. Stav 0 reprezentuje deficit zásob a stav K plný sklad. Časová stabilizácia tohto systému je vďaka možnosti odmietania dodávok nepodmienená – zaťaženie systému $\rho = \frac{\lambda}{\mu}$ môže byť aj väčšie než 1.

Z prechodového grafu môžeme odvodiť stacionárne rozdelenia stavov systému $(\pi_0, \pi_1, \ldots, \pi_K), \pi_j = A\rho^j, A = (1-\rho)/(1-\rho^{K+1})$. Aj tu *priemernú výšku zásob v sklade* odhadneme stredným počtu zákazníkov v systéme

$$\bar{Q} = \sum_{j=0}^{K} j\pi_j = \frac{\rho}{1-\rho} \cdot \frac{1-(K+1)\rho^K - K\rho^{K+1}}{1-\rho^{K+1}} \ . \tag{4.55}$$

Nákladová funkcie $\mathcal{H}_8(\rho)$ obsahuje okrem priemerných jednotkových nákladov skladovania a priemerných jednotkových nákladov deficitu aj priemerné jednot-

kové straty z nerealizovaných predajov, resp. penalizáciu neprebratých dodávok so stratami C_P na jednotku zásob.

$$\begin{aligned}\mathcal{H}_8(\rho) &= C_S\bar{Q} + C_D\lambda\pi_0 + C_P\lambda\pi_K \\ &= C_S\Big(\frac{\rho}{1-\rho}\cdot\frac{1-(K+1)\rho^K - K\rho^{K+1}}{1-\rho^{K+1}}\Big)\frac{C_D\lambda(1-\rho)}{1-\rho^{K+1}} + \frac{C_P\lambda\rho^K(1-\rho)}{1-\rho^{K+1}}\ .\end{aligned}$$

Optimálne riešenie ρ^*, ktoré minimalizuje jednotkovú nákladovú funkciu $\mathcal{H}_8(\rho)$, už nevieme vo všeobecnom prípade vyjadriť vzorcom, a tak je potrebné ho hľadať numericky. Do úvahy tu prichádza v najhoršom prípade aj úplná enumerácia funkčných hodnôt po vhodnej zvolenej diskretizácii definičného oboru cieľovej funkcie.

Netriviálny je už krajný prípad modelu – $M/M/1/2$, kde sa jedna jednotka zásob môže skladovať počas doby vyskladňovania predchádzajúcej jednotky zásob. Nákladová funkcia tu má tvar

$$\mathcal{H}_8(\rho) = \frac{1}{(1-\rho)(1-\rho^3)}\Big(C_S\cdot\rho(1-3\rho^2-2\rho^3) + C_D\cdot(1-\rho)^2 C_P\cdot\rho^2(1-\rho)^2\Big)\ .$$

Ani tento prípad nevieme vo všeobecnosti riešiť analyticky.

V situácii, keď nie je kapacita skladu K vopred daná, má zmysel otázka *ekonomickej kapacity K^* skladu*. Pri dodatočných *jednotkových nákladoch za skladovacie miesto za jednotku času* C_M dostávame jednotkovú nákladovú funkciu $\mathcal{H}_9(\rho, K)$ dvoch neznámych ρ a K ($\rho > 0$ a $K > 1$ je prirodzené číslo)

$$\mathcal{H}_9(\rho, K) = C_M\cdot K + C_S\cdot\bar{Q} + C_D\lambda\cdot\pi_0 + C_P\cdot\lambda\pi_K\ ,$$

kde aj charakteristiky systému $\bar{Q}, \pi_0, \pi_K$ treba chápať ako funkcie neznámych parametrov ρ a K. Pri numerickej minimalizácii funkcie $\mathcal{H}_9(\rho, K)$ môžeme postupovať tak, že pre fixované hodnoty premennej $K = 2, 3, \ldots$ vypočítame optimálnu záťaž systému $\rho(K)$ a potom vyberieme najlacnejšie riešenie ρ^* a K^*

$$\mathcal{H}_9(\rho^*, K^*) = \min_{\rho(K)>0, K=2,3,\cdots}\mathcal{H}_9(\rho(K), K)\ .$$

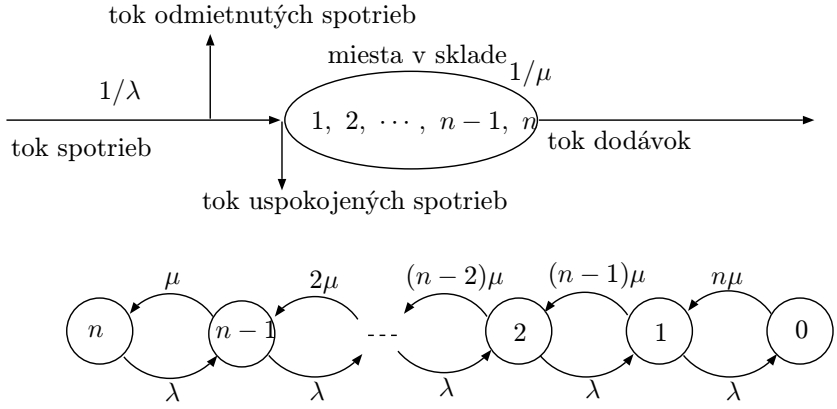
V modeloch so vstupným tokom dodávok môžeme prirodzene použiť aj zložitejšie nemarkovovské jednolinkové frontové systémy G/G/1. Dobrým kompromisom medzi všeobecnosťou modelu a praktickými charakteristikami režimu zásobovania ponúkajú semimarkovove frontové systémy $E_r/E_q/1$, v ktorých môžeme realisticky modelovať aj rozptyl dodávkového cyklu, resp. rozptyl doby

vyskladnenia zásob. Čiastočne tak môžeme riešiť i problematický predpoklad jednotkovej dodávky rovnej spotrebovávanej jednotke zásoby, keď počet fáz r interpretujeme ako veľkosť dodávky a počet fáz q ako veľkosť jednotkovej spotreby. Frontové modely založené na dávkových systémoch $M^X/M^Y/1$ [8] ponúkajú (za cenu podstatne náročnejšieho matematického aparátu) aj možnosť modelovať stochastiky veľkosť dodávky, resp. veľkosť spotrebnej dávky. Ako uvidíme ďalej, táto poznámka sa týka aj modelov so vstupným tokom spotreby.

4.4.2 Modely so vstupným tokom spotrieb

Doteraz sme modelovali prevádzku skladov systémami hormadnej obsluhy, v ktorých vstupný tok zákazníkov predstavoval dodávky tovarov do skladu. V modeloch, ktoré budeme študovať v tomto odseku, zvolíme iný prístup. Pri tomto prístupe vstupný tok zákazníkov bude reprezentovať požiadavky spotreby komodít zo skladu.

Najskôr uvažujme jednoduchý **frontový inverzný model** $M/M/n/n$ **so vstupným tokom spotrieb** , schéma a prechodový graf je na obr. 4.11. Tok spotrieb je modelovaný Poissonovým tokom s intenzitou λ. Dĺžka medzery



Obr. 4.11: Schéma a prechodový graf inverzného systému $M/M/n/n$

medzi príchodmi jednotkových požiadaviek do skladu má strednú hodnotu $\frac{1}{\lambda}$. Doba „obsluhy“ systému je *dobou dodacej lehoty* na novú jednotku skladovanej

[8] Dávkami sú celočíselné náhodné premenné X a Y.

komodity a má exponenciálne rozdelenie s parametrom μ. Jednotka spotreby opúšťa vždy systém okamžite po svojom príchode. Ak však nájde nejakú linku voľnú (neaktívnu), potom je v sklade aspoň jedna jednotka zásob, ktorou je táto požiadavka uspokojená a zároveň je vystavená nová objednávka na doplnenie zásob skladu. Počet voľných liniek teda zodpovedá momentálnej veľkosti zásob v sklade. Parameter n reprezentuje *kapacitu skladu* – maximálny počet jednotiek skladovanej komodity. Po strednej dobe $\frac{1}{\mu}$ je takto k dispozícii nová skladovaná jednotka. Ak požiadavka spotreby nájde všetky linky aktívne, znamená to, že sklad je prázdny, a preto je v dôsledku deficitu zásob odmietnutá.

Markovovský reťazec tu má množinu stavov $S = \{0, 1, 2, \ldots, n\}$, stav i udáva *veľkosť zásob v sklade*, resp. $n - i$ udáva *počet dodávok na ceste*, a tak stav 0 reprezentuje stav deficitu zásob, keď nemôže byť žiadna jednotka spotreby uspokojená. Na rozdiel od klasického systému s odmietaním M/M/n/n, v *inverznom systéme* však príchod zákazníka (jednotky spotreby) nespôsobuje zväčšenie stavu systému o jednotku, ale naopak zmenšenie stavu systému o jednotku, čomu zodpovedá prechodový graf na obr. 4.11.

Reťazec má jediné stacionárne rozdelenie stavov systému $(\pi_0, \pi_1, \ldots, \pi_n)$, ktoré opäť môžeme ľahko vypočítať grafovou metódou v tvare

$$\pi_j = \begin{cases} \dfrac{\alpha^{n-j}}{(n-j)!}\pi_n & \text{ak } 0 \leq j \ < n \\ \left(\displaystyle\sum_{j=0}^{n} \frac{\alpha^{n-j}}{(n-j)!}\right)^{-1} & \text{ak } j = 0 \end{cases} , \tag{4.56}$$

pričom parameter $\alpha = \frac{\lambda}{\mu}$ sa nazýva *predstih zásob*. Potom priemerný stav zásob $\bar{Q}$ v stabilizovanom systéme je určený vzťahom

$$\bar{Q} = \sum_{j=1}^{n} j\pi_j = \pi_n \sum_{j=1}^{n} \frac{j\alpha^{n-j}}{(n-j)!} . \tag{4.57}$$

Pre výpočet charakteristík $\pi_0, \pi_n, \bar{Q}$ možno výhodne využiť tabelované *funkcie kumulovaných hodnôt Poissonovho rozdelenia*

$$P_n(x) = \sum_{j=0}^{n} \frac{x^j}{j!} e^{-x} ,$$

ktoré vedú k vzťahom

$$\pi_0 = 1 - \frac{P_{n-1}(\alpha)}{P_n(\alpha)} \ , \tag{4.58}$$

$$\pi_n = \frac{e^{-\alpha}}{P_n(\alpha)} \ , \tag{4.59}$$

$$\bar{Q} = \frac{nP_n(\alpha) - \alpha P_{n-1}(\alpha)}{P_n(\alpha)} \ . \tag{4.60}$$

Priemerný počet objednaných jednotiek (dodávok) je zrejme $n - \bar{Q}$. Optimálnu hodnotu kapacity skladu n^* možno numericky vypočítať maximalizáciou *priemerného čistého zisku za časovú jednotku*

$$\mathcal{Z}_1(n) = C_Z \cdot (n - \bar{Q}) - C_S \cdot \bar{Q} - C_D \cdot \lambda\pi_0 \ , \tag{4.61}$$

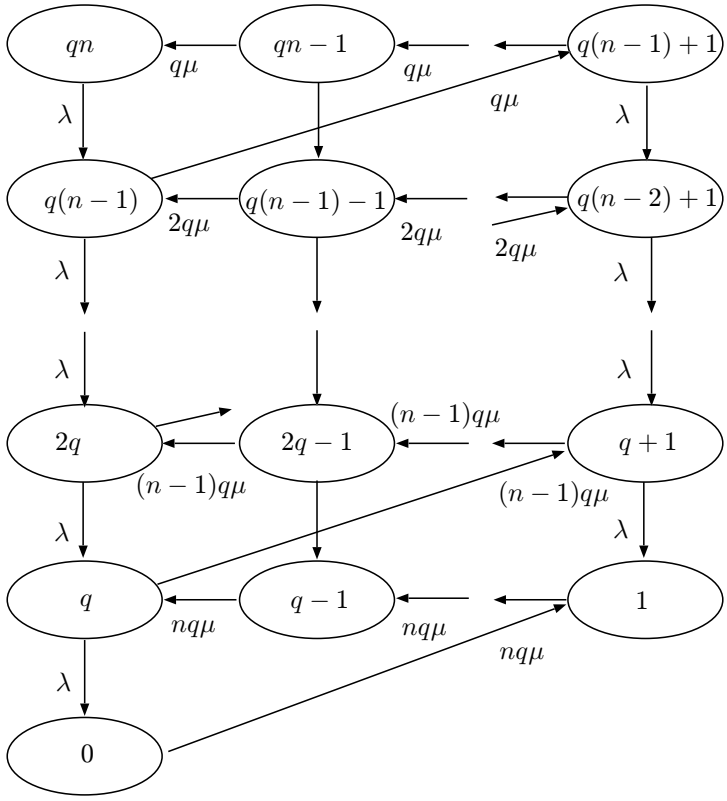
kde C_Z je hrubý zisk z predaja jednej jednotky zásob, C_S sú náklady na skladovanie jednotky zásob za časovú jednotku a C_D sú náklady na jednotku deficitu zásob.

Doteraz sme pri frontových modeloch predpokladali, že dodávka do skladu je tvorená jednotkou zásoby rovnej jednotke spotreby. Ak spotreba vždy požaduje jednorazovo q jednotiek zásob, potom tento problém rieši **frontový inverzný model** $M/E_q/n/n$ **so vstupným tokom spotrieb** s prechodovým grafom na obr. 4.12. Tok spotrieb je opäť modelovaný Poissonovým tokom s intenzitou λ jednotiek za časovú jednotku.

Doba dodacej lehoty na q jednotiek zásob má q-fázové Erlangovo rozdelenie so strednou hodnotou $1/\mu$. Takže ak je v sklade dostatočná zásoba, potom q jednotiek zásob sa jednorazovo spotrebuje po svojom príchode a zároveň je vystavená nová objednávka na doplnenie zásob o q jednotiek. V opačnom prípade nie je spotreba (ani čiastočne) uspokojená. Kapacita skladu je tu nq jednotiek. Po strednej dobe $\frac{1}{\mu}$ od objednania dodávky je takto k dispozícii dalších q nových jednotiek. Predpoklad erlangovskej dodacej lehoty umožňuje aj predstavu q jednotkových dodávok s exponenciálnou dodacou lehotou a strednou hodnotou $\frac{1}{q\mu}$.

Z prechodového grafu na obr. 4.12 priamo zostrojíme $(nq+1) \times (nq+1)$ maticu intenzít $\Omega, (\omega_{01} = nq\mu, \omega_{q0} = \lambda, \dots)$ a vypočítame jediné stacionárne rozdelenie stavov markovovského reťazca $\vec{\pi} = (\pi_0, \pi_1, \dots, \pi_{nq})$ riešiac systém[9]

[9]Každá rovnica homogénnej časti systému (4.62) je lineárnou kombináciou ostatných, a tak ľubovoľná z nich môže byť nahradená normalizačnou podmienkou.



Obr. 4.12: Prechodový graf inverzného systému $M/E_q/n/n$

lineárnych rovníc

$$\vec{\pi}\Omega = 0,\ \sum_{i \in S} \pi_j = 1, \tag{4.62}$$

s množinou stavov $S = \{0, 1, \ldots, nq\}$.
Potom priemerný stav zásob $\bar{Q}$ v stabilizovanom systéme je určený vzťahom

$$\bar{Q} = \sum_{j \in S} j\pi_j, \tag{4.63}$$

a priemerný počet objednaných jednotiek zásob je $nq - \bar{Q}$. Funkciu *priemerného čistého zisku za časovú jednotku* získame modifikáciou vzťahu (4.61) v tvare

$$\mathcal{Z}_2(n,q) = C_Z \cdot (nq - \bar{Q}) - C_S \cdot \bar{Q} - C_D \cdot \lambda \sum_{j=0}^{q-1} \pi_j \ . \tag{4.64}$$

Optimálnu hodnotu kapacity skladu n^*q^* numericky určíme maximalizáciou funkcie $Z(n,q)$ pre kladné prirodzené čísla n, q. Naviac n^* udáva maximálny počet zásob na ceste a q^* ekonomickú veľkosť spotrebnej dávky. V praktických úlohách môžeme očakávať prirodzené horné hranice na premenné n, q, čo však vedie len k rovnako obťažnému viazanému extrému funkcie $Z(n,q)$.

Ak je však známe diskrétne pravdepodobnostné rozdelenie spotreby Y, potom tento problém rieši všeobecnejší **frontový inverzný model** $M/M^Y/n/n$ **so vstupným tokom spotrieb**, s pomerne náročným výpočtom matice intenzít Ω a analogickým výpočtom (4.62) stacionárneho rozdelenia stavov systému.

4.5 Zásobovanie ako hra

Proces zásobovania nejakou komoditou spravidla nie je len problém skladu, ale viacerých subjektov reťazca výroba – sklad – spotreba, v ktorom každý sleduje svoj úžitok. Konfliktné situácie, ktoré tu vznikajú, možno modelovať pomocou modelov teórie hier.

4.5.1 Model hry proti prírode

Spotrebu môžeme chápať aj ako známy (riziko), resp. neznámy (neurčitosť) náhodný mechanizmus – neinteligentný hráč, ktorý nesleduje žiaden kvantifikovateľný úžitok, a tak jediným inteligentným hráčom je tu sklad.

Príkladom takejto *hry proti prírode pri riziku* je m**-komoditný rizikový** (h,Q) **model so stratenou spotrebou**, ktorý vznikne z už prezentovaného stochastického (h,Q) modelu so stratenou spotrebou, uvažujúci s pravdepodobnosťou ψ_i dodávku a následnú spotrebu i-tej komodity a intenzitou spotreby λ_i. Predpokladá sa totiž, že nie je vopred známe, ktorá komodita bude po objednaní dodaná na sklad, len jej pravdepodobnosť. Pre jednoduchosť budeme predpokladať, že $\lambda_1 < \lambda_2 < \cdots < \lambda_m$.

Priestor stratégií prvého hráča - *skladu*, je množina

$$\mathcal{X} = \{(h,Q) \mid (h,Q) \in (0,\infty) \times (0,\infty),\ h < Q\}$$

prípustných dvojíc hodnôt veľkosti dodávky Q a hladiny objednania h. Priestor stratégií druhého hráča *spotreby*, je potom množina

$$\mathcal{Y} = \{\lambda_1, \lambda_2, \ldots \lambda_m\}$$

prípustných intenzít spotreby. V tejto hre má zmysel výplatná funkcia[10] prvého hráča

$$\mathcal{M}(h, Q, \lambda) = C_N \lambda - (C_A + C_S \cdot \bar{Q}_K(h, Q, \lambda) + C_D \cdot \bar{U}(h, \lambda))/\bar{C}(h, Q, \lambda)$$

udávajúca *priemerný ročný zisk skladu pre komoditu s intenzitou* λ. Priemerná dĺžka dodávkového cyklu $\bar{C}(h, Q, \lambda)$ je tu určená vzťahom (4.25). Podobne aj priemerný deficit zásob $\bar{U}(h, \lambda)$ je definovaný vzťahom (4.24) a priemerný stav zásob $\bar{Q}_K(h, Q, \lambda)$ vzťahom (4.27). Vidíme, že sú naviac funkciami aj rozhodovacej premennej λ.

Výplatnú funkciu spotreby – druhého hráča – môžeme vynechať, nakoľko je pre tohto neinteligentného hráča nevýznamná, nerozhoduje sa podľa nej, a tak ani pre prvého hráča nenesie žiadnu informáciu.

Optimálna stratégia prvého hráča (h^*, Q^*) v tejto hre $\big(\mathcal{X}, \mathcal{Y}, \mathcal{M}(h, Q, \lambda)\big)$ je potom určená vzťahom

$$(h^*, Q^*) = arg \max_{(h,Q)\in\mathcal{X}} \sum_{\lambda_i \in Y} \mathcal{M}(h, Q, \lambda_i)\psi_i,$$

v ktorom súčet vážených výhier pri čistých stratégiách druhého hráča udáva *strednú hodnotu hry* pri stratégii prvého hráča (h, Q).

V prípade, keď nepoznáme ani hodnoverné odhady početnosti dodávok komodít, t. j. nepoznáme pravdepodobnosti $\psi_1, \psi_2, \ldots, \psi_m$, má zmysel ***m*-komoditný neurčitý (h, Q) model so stratenou spotrebou**, ktorý môžeme formulovať ako *hru proti prírode pri neurčitosti* v tvare $\big(\mathcal{X}, \mathcal{Y}, \mathcal{M}(h, Q, \lambda)\big)$. Definície optimálnych stratégií prvého hráča je zvykom nazývať *princípmi optimality*, do úvahy prichádzajú hlavne tieto:

- *Princíp nedostatočnej evidencie* považuje všetky stratégie spotreby $\mathcal{Y}$ (druhého hráča) za rovnako pravdepodobné, a tak optimálna stratégia skladu (h_1^*, Q_1^*) je rovná

$$(h_1^*, Q_1^*) = arg \max_{(h,Q)\in\mathcal{X}} \sum_{\lambda_i \in Y} \mathcal{M}(h, Q, \lambda_i)/m.$$

[10] Kladná hodnota výplatnej funkcie zodpovedá výhre, záporná prehre hráča.

- *Princíp minimaxu* predpokladá voľbu takej stratégie spotreby, ktorá vedie k najvýhodnejšiemu možnému výsledku. Sklad volí stratégiu (h_2^*, Q_2^*) maximalizujúcu zaručenú výhru v $\mathcal{X}$

$$(h_2^*, Q_2^*) = arg \max_{(h,Q)\in\mathcal{X}} \min_{\lambda_i\in\mathcal{Y}} \mathcal{M}(h, Q, \lambda_i) \ .$$

- *Princíp stredného optimizmu* volí kompromis medzi krajným optimizmom a krajným pesimizmom voľbou stratégie skladu (h_3^*, Q_3^*) definovanej vzťahom

$$(h_3^*, Q_3^*) = arg \max_{(h,Q)\in\mathcal{X}} \left(\max_{\lambda_i\in\mathcal{Y}} \mathcal{M}(h, Q, \lambda_i) - \min_{\lambda_i\in\mathcal{Y}} \mathcal{M}(h, Q, \lambda_i) \right) \ .$$

Uvedené tri princípy optimality môžeme využiť aj pri tvorbe takých žiadaných *scenárov dodávky* $(Q_O, Q_K, Q_P), Q_O \geq Q_K \geq Q_P$, kde Q_O je *optimistická*, Q_K *kompromisná* a Q_P *pesimistická* veľkosť dodávky na sklad.

4.5.2 Model nekooperatívnej hry

Uvažujme **model oligopolu** [20], kde oligopolom rozumieme trh daného počtu oligopolistov (výrobcov) snažiacich sa predať jeden druh výrobkov, pričom oligopolisti nesmú medzi sebou utvárať žiadne separátne dohody, teda ani o veľkosti objemu výroby.

Ďalej sa pre jednoduchosť obmedzíme na **kapacitne obmedzený Wilsonov model oligopolu**, v ktorom priemerné denné[11] náklady $\mathcal{H}_i(Q_i)$ na objem výroby Q_i každého oligopolistu $i \in \{1, 2, \ldots, N\}$ sú určené analogicky ako vo Wilsonovom modeli zásob vzťahom (4.3)

$$\mathcal{H}_i(Q_i) = C_{iV}\frac{\lambda_i}{Q_i} + C_{iS}\frac{Q_i}{2}, \ K_{iA} \leq Q_i \leq K_{iB} \ ,$$

pričom i-ty oligopolista je určený nasledujúcimi charakteristikami:

- λ_i intenzitou výroby – priemerným objemom výroby za deň,
- C_{iV} priemernými výrobnými nákladmi na jednotku výrobku,
- C_{iS} priemernými skladovacími nákladmi výroby na jednotku výrobku za deň,

[11] Pevnou časovou jednotkou obdobia sa spravidla uvádza deň a nie rok vzhľadom na štatistikou vykazovanú dennú produkciu oligopolistov.

- K_{iA} minimálnou dennou produkciou – nutnou kapacitou výroby za deň,
- K_{iB} maximálnou dennou produkciou – možnou hraničnou kapacitou výroby za deň.

Výrobné náklady na výrobu Q_i jednotiek sú u i-teho oligopolistu určené konvexnou funkciou $c_i(Q_i)$. Trh je tu zas charakterizovaný konkávnou klesajúcou funkciou $f(Q)$ udávajúcou *cenu výrobku pri celkovom množstve výroby* Q.

Treba určiť úrovne výroby $(Q_1, Q_2, \ldots, Q_N)$ tak, aby každý oligopolista s prihliadnutím k ostatným maximalizoval svoj denný čistý zisk (príjem z predaja zmenšený o náklady). Dostávame tak *nekooperatívnu hru N hráčov (oligopolistov) v normálnom tvare*

$$\left(\mathcal{X}_1, \mathcal{X}_1, \ldots, \mathcal{X}_N, \mathcal{M}_1(\cdot), \mathcal{M}_2(\cdot), \ldots, \mathcal{M}_N(\cdot)\right) , \tag{4.65}$$

s priestormi stratégií $\mathcal{X}_i = \langle K_{iA}, K_{iB} \rangle$ a výplatnými funkciami

$$\mathcal{M}_i(Q_1, Q_2, \ldots, Q_N) = Q_i \cdot f(\sum_{i=1}^{N} Q_i) - \mathcal{H}_i(Q_i) - c_i(Q_i) \tag{4.66}$$

pre i-teho hráča.

Cieľom hry (4.65) je nájsť *rovnovážnu stratégiu hry* $(Q_1^*, Q_2^*, \ldots, Q_N^*)$ požadujúcu pre každého hráča $i \in \{1, 2, \ldots, N\}$ a ľubovoľný objem výroby $Q_i \in \mathcal{X}_i$ splnenie vzťahu

$$\mathcal{M}_i(Q_1^*, \ldots, Q_{i-1}^*, Q_i, Q_{i+1}^*, \ldots, Q_N^*) \leq \mathcal{M}_i(Q_1^*, , \ldots, Q_{i-1}^*, Q_i^*, Q_{i+1}^*, \ldots, Q_N^*) ,$$

ktorý garantuje i-tému hráčovi maximálnu výhru, pokiaľ sa ostatní hráči od svojej rovnovážnej stratégie neodchýlia. Rovnovážnu stratégia však už nič nehovorí o priebehu hry, ak sa od nej odchýlia dvaja (prípadne viacerí) hráči.

Ak máme pre každé $i \in \{1, 2, \ldots, N\}$ výplatnú funkciu $\mathcal{M}_i(\ldots, Q_i, \ldots)$ v premennej Q_i nielen konkávnu, ale aj diferencovateľnú, potom možno ukázať, že $(Q_1^*, Q_2^*, \ldots, Q_N^*)$ je rovnovážnou stratégiou hry, ak platí vzťah

$$\lim_{Q_i \to Q_i^*} \frac{\partial \mathcal{M}_i(Q_1^*, \ldots, Q_{i-1}^*, Q_i, Q_{i+1}^*, \ldots, Q_N^*)}{\partial Q_i} \begin{cases} = 0 & \Rightarrow \quad K_{iA} \leq Q_i^* \leq K_{iB} \\ > 0 & \Rightarrow \quad Q_i^* = K_{iB} \\ < 0 & \Rightarrow \quad Q_i^* = K_{iA} \end{cases}$$

ktorý umožňuje jej hľadanie *gradientnou metódou*. V mnohých prípadoch však konverguje aj nasledujúcu jednoduchšia iteračná ε-metóda:

Algoritmus 4.2. Iteračná ε-metóda

Krok 1: Definujeme N-ticu $\mathbf{Q^1} = (Q_1^1, \cdot, Q_N^1)$ so zložkami $Q_i^1 = \sqrt{\frac{2\lambda_i C_{iV}}{C_{iS}}}$.

Krok 2: Postupne pre všetky indexy $i = 1, \ldots, N$ nech je Q_i^2 riešením rovnice

$$g_i(Q_i) = \frac{\partial \mathcal{M}_i(Q_1^1, \ldots, Q_{i-1}^1, Q_i, Q_{i+1}^1, \ldots, Q_N^1)}{\partial Q_i} = 0$$

a definujeme zložky N-tice $\mathbf{Q^3}$ takto:

$$Q_i^3 = \begin{cases} Q_i^2 & \text{ak } g_i(Q_i^2) = 0 \text{ a} \quad K_{iA} \le Q_i^2 \le K_{iB} \\ K_{iB} & \text{ak } g_i(Q_i^2) > 0 \text{ alebo} \quad Q_i^2 > K_{iB} \\ K_{iA} & \text{ak } g_i(Q_i^2) < 0 \text{ alebo} \quad Q_i^2 < K_{iA} \end{cases}$$

Krok 3: Ak $\sum_{i=1}^{N} \left(|Q_i^1 - Q_i^3| < \varepsilon\right) = 0$, potom *STOP* s riešením $\mathbf{Q^*} = \mathbf{Q^1}$. Ináč položíme $\mathbf{Q^1} = \mathbf{Q^3}$ a **GOTO Krok 2**.

4.5.3 Model kooperatívnej hry

Ak medzi oligolipolistami je kooperácia zakázaná, pri vzťahu medzi dodávateľom a obchodným reťazcom je žiaduca.

Uvažujme nasledujúcu *kooperatívnu hru medzi dodávateľom a obchodným reťazcom*. Monopolný dodávateľ zásobuje obchodný reťazec (ďalej len reťazec) komoditou za nákupnú cenu C_N na jednotku komodity. Reťazec ju predáva za maloobchodnú cenu p, ktorú treba určiť, budeme predpokladať, že $p \ge C_N$.

Predpokladajme, že intenzitu spotreby reťazca možno v zhode s prácou [31] štatisticky odhadnúť pomocou klesajúcej funkcie maloobchodnej ceny komodity v tvare

$$\lambda(p) = \alpha p^{-\beta}, \quad \alpha > 0, \quad 0 < \beta < 1 \ . \tag{4.67}$$

Dodávateľ nemá vlastný sklad, ale reťazec má vlastný sklad. A preto si delia skladovacie náklady tak, že jednotkové skladovacie náklady dodávateľa sú konštantné C_S, zatiaľčo v reťazci sú lineárnou funkciou maloobchodnej ceny komodity, t. j. $R_S \cdot p$. Reťazec nedisponuje vlastným vozidlovým parkom, takže akvizičné náklady C_A za dodávku veľkosti Q, ktorú treba určiť, si uplatňuje dodávateľ voči reťazcu. Dodávateľ je tiež zaťažený nákladmi za dodávku C_D od výrobcu.

Ak použijeme *základný Wilsonov model zásob* pre dodávateľa a *Wilsonov model zásob s obstarávaním*, pre reťazec dostávame zo vzťahov (4.3) a (4.7)

priemerné ročné náklady $\mathcal{H}_D$ a $\mathcal{H}_R$ ako nasledujúce funkcie maloobchodnej ceny a veľkosti dodávky:

$$\mathcal{H}_D(p,Q) = \alpha C_D \frac{p^{-\beta}}{Q} + \frac{Q}{2} C_S \ , \tag{4.68}$$

$$\mathcal{H}_R(p,Q) = \alpha C_A \frac{p^{-\beta}}{Q} + \frac{Q}{2} R_S p + C_N \alpha p^{-\beta} \ . \tag{4.69}$$

Zisk dodávateľa a reťazca sú potom postupne rovné

$$\begin{aligned} \mathcal{M}_D(p,Q) &= C_N \alpha p^{-\beta} - \mathcal{H}_D(p,Q) \ , \\ \mathcal{M}_R(p,Q) &= \alpha p^{1-\beta} - \mathcal{H}_R(p,Q) \ . \end{aligned}$$

Predpokladajme, že reťazec je ochotný investovať čiastku nanajvýš C_0. Dostávame tak nasledujúcu hru dvoch hráčov – dodávateľa a reťazec – v normálnom tvare:

$$\begin{aligned} \big(\mathcal{X}, \mathcal{Y}, \mathcal{M}_D(p,Q), \mathcal{M}_R(p,Q)\big), \quad \mathcal{X} &= \{Q \mid Q \geq 0\}, \\ \mathcal{Y} &= \{p \mid p \geq C_N, \mathcal{M}_R(p,Q) \leq C_0\} \end{aligned}$$

Potom má zmysel riešiť optimalizačnú úlohu (UKDR):

$$\max \mathcal{M}_D(p,Q) + \mathcal{M}_R(p,Q) \tag{4.70}$$

$$\mathcal{H}_R(p,Q) \leq C_0 \tag{4.71}$$

$$p \geq C_N, \ Q \geq 0 \tag{4.72}$$

Cieľová funkcia (4.70) predstavuje celkový zisk dodávateľa a reťazca, ktorý sa maximalizuje. Podmienka (4.71) pokrýva náklady spojené s nákupom a skladovaním komodity. Podmienky (4.72) zabezpečujú prípustnú voľbu maloobchodnej ceny p a veľkosti dodávky p.

Nech (p^*, Q^*) sú optimálne stratégie z riešenia úlohy UKDR. Dodávateľ i reťazec by sa teda mohli dohodnúť na dodávkach veľkosti Q^*, ktoré reťazec nakúpi za cenu C_N a predá za cenu p^*. Môže však vzniknúť pochybnosť, či je výhodné pre reťazec koordinovať voľbu maloobchodnej ceny i veľkosti dodávky s dodávateľom.

Garantovaný zisk reťazca môžeme dostať z riešenia nasledujúcej optimalizačnej úlohy (UGZR):

$$\max \mathcal{M}_R(p, Q) \tag{4.73}$$

$$Q = \sqrt{\frac{2\alpha p^{-\beta} C_A}{R_S p}} \tag{4.74}$$

$$\mathcal{H}_R(p, Q) \leq C_0 \tag{4.75}$$

$$p \geq C_N \tag{4.76}$$

Cieľovou funkciou (4.73) je zisk reťazca, ktorý sa maximalizuje. Rovnica (4.71) volí optimálnu veľkosť dodávky vo Wilsonovom modeli s obstarávaním. Obmedzenie (4.72) opäť ohraničuje náklady spojené s nákupom a skladovaním komodity. Obligatórna podmienka (4.76) zabezpečuje prípustnú voľbu maloobchodnej ceny komodity. Ak je p_R^* optimálnym riešením úlohy, potom po dosadení do vzťahu (4.71) dostávame príslušnú optimálnu veľkosť dodávky Q_R^*.

Nech (p_R^*, Q_R^*) sú optimálne stratégie z riešenia úlohy UGZR a (p^*, Q^*) sú optimálne stratégie z riešenia úlohy UKDR. Ak platí podmienka

$$\mathcal{M}_R(p_R^*, Q_R^*) \leq \mathcal{M}_R(p^*, Q^*) \ ,$$

potom sa reťazcu oplatí prijatím riešenia (p^*, Q^*) koordinovať svoju stratégiu s dodávateľom. V opačnom prípade je pre reťazec výhodné trvať na dodávke veľkosti Q_R^* alebo sa dohodnúť s dodávateľom na dodávke veľkosti Q^* a doplatku[12] vo výške aspoň $\mathcal{M}_R(p^*, Q^*) - \mathcal{M}_R(p_R^*, Q_R^*)$.

Obe optimalizačné úlohy sú úlohami nelineárneho matematického programovania. Pri ich riešení možno použiť napr. *Solver – Riešiteľ* implementovaný v tabuľkovom procesore *Excel*. Vyžaduje však pomerne citlivú voľbu jeho parametrov, v opačnom prípade nenájde riešenie s odvolaním sa na nedostatočný počet iteračných krokov.

4.6 Rovnomerné zásobovanie skladov

Pri zásobovaní viacerých skladov firiem (napr. obchodných reťazcov) vozidlami z viacerých vozidlových parkov výrobcov, resp. špedičných firiem vzniká problém vytvárania takého rozvrhu prác vozidiel, pri ktorom sú sklady reťazcov zásobované čo najrovnomernejšie.

[12] V tomto prípade dodávateľ a reťazec hrajú kooperatívnu hru s prenosom výhry.

Najskôr uvažujme **základný model rovnomerného zásobovania skladov**. Nech je daná množina m skladov a množina n vozidiel. Každé vozidlo môže v danom období vykonať niekoľko rozvozných trás typu zdroj → sklad → zdroj. Predpokladajme, že prvky danej matice A udávajú jednotkové množstvá nejakej komodity prepravovanej z nejakého zdroja do i-teho skladu rozvoznou trasou j-teho vozidla. Celková zásoba i-teho skladu vytvorená dodávkami vozidiel je $s_i = \sum_{j=1}^{n} a_{ij}$. Ak existujú veľké rozdiely medzi takto vytvorenými zásobami skladov $s_1, s_2, \cdots, s_m$, skladníci budú proti tomu protestovať. Cieľom je nájsť také permutácie dodávok v stĺpcoch matice A, pri ktorých sú kumulované zásoby v skladoch čo najrovnomernejšie.[13]

Problémom permutácií matice MPP (matrix permutation problem) rozumieme nasledujúcu úlohu: Nech sú $M = \{1, 2, \ldots, m\}$ a $N = \{1, 2, \ldots, n\}$ konečné neprázdne množiny indexov danej reálnej matice $\mathrm{A} = (a_{ij}) \in \mathbb{R}^{m \times n}$. Hľadá sa sa taká usporiadaná n-tica $\psi = (\psi_1, \psi_2, \ldots, \psi_n)$ permutácií stĺpcov M matice A, pre ktorú je

$$F(s_1, s_2, \ldots, s_m) \quad \rightarrow \quad \min \tag{4.77}$$

$$s_i \quad = \quad \sum_{j \in N} a_{\psi_{ij}, j} \qquad i \in M, \tag{4.78}$$

$$\psi_j \quad = \quad (\psi_{1j}, \psi_{2j}, \ldots, \psi_{mj}) \in \Pi(N) \qquad j \in N, \tag{4.79}$$

kde $F(s_1, s_2, \ldots, s_m)$ je nejaká miera nerovnomernosti $(s_1, s_2, \ldots, s_m) \in \mathbb{R}^n$ a $\Pi(N)$ je množina všetkých permutácií na množine N.

Poznamenajme, že nulové prvky $a_{ij} = 0$ zodpovedajú „fiktívnym dodávkam“ j-teho vozidla i-temu skladu. Formulácia MPP pripúšťa aj záporné prvky matice, ktoré nám umožňujú modelovať prípadné vopred požadované rozdiely v celkovej veľkosti zásob v niektorých skladoch.

Pre každé riešenie ψ úlohy MPP definované maticou A tak dostávame permutovanú maticu A^{ψ}

$$\mathrm{A}^{\psi} = \begin{pmatrix} a_{\psi_{11},1} & a_{\psi_{12},2} & \ldots & a_{\psi_{1n},n} \\ a_{\psi_{21},1} & a_{\psi_{22},2} & \ldots & a_{\psi_{2n},n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{\psi_{m1},1} & a_{\psi_{m2},2} & \ldots & a_{\psi_{mn},n} \end{pmatrix},$$

ktorá udáva dodávky zásob jednotlivými vozidlami. Jej riadkové súčty zodpovedajú rozdeleniu zásob v skladoch $(s_1^{\psi}, s_2^{\psi}, \ldots, s_m^{\psi})$.

[13] Úloha má takúto zábavnú interpretáciu ako úloha o krívajúcej stonožke: Máme k dispozícii články nôh stonožky. Každá z nôh má daný počet navzájom nezameniteľných druhov článkov. Rôzne druhy článkov treba poskladať tak, aby stonožka čo najmenej krívala.

V prácach [53], [52] bolo dokázané, že ak je mierou nerovnomernosti cieľovej funkcie $F(s_1, s_2, \ldots, s_n)$ nejaká *Shur-konvexná* funkcia[14], potom je MPP NP-ťažký problém. Príkladmi takýchto funkcií sú

$$F_{max}(s_1, s_2, \ldots, s_m) = \max_{i \in M} s_i, \tag{4.80}$$

$$F_{dif}(s_1, s_2, \ldots, s_m) = \max_{i \in M} s_i - \min_{i \in M} s_i, \tag{4.81}$$

$$F_{sqr}(s_1, s_2, \ldots, s_m) = \sum_{i \in M} (s_i - \overline{s})^2, \ \overline{s} = \frac{1}{m} \sum_{i \in M} s_i. \tag{4.82}$$

Cieľová funkcia určená mierou (4.80) potom minimalizuje zásoby sklad s najväčšími zásobami. Pri miere (4.81) ide o rozdiel medzi najväčšou a najmenšou zásobou v skladoch a pri miere (4.82) o súčet štvorcov rozdielov zásob v skladoch a „ideálnej“ priemernej zásoby.

Možno ukázať, v zhode s očakávaním, že miera nerovnomernosti F_{sqr} je jemnejšia než F_{dif} a tá je jemnejšia než F_{max}.

Úloha MPP má aj nasledujúcu interpretáciu, nazveme ju **kalendárny model rovnomerného zásobovania skladu**. Nech je daná množina m vozidiel a množina n pracovných dní. Každé vozidlo môže v tomto období vykonať nanajvýš jednu rozvoznú trasu typu zdroj → sklad → zdroj. Predpokladajme, že prvky danej matice A udávajú jednotkové množstvá nejakej komodity prepravovanej z nejakého zdroja i-tym vozidlom skladu rozvoznou trasou j-ty deň obdobia. Celkový výkon i-teho vozidla vytvorený dodávkami v uvažovanom období n dní je $s_i = \sum_{j=1}^{n} a_{ij}$. Ak existujú veľké rozdiely medzi takto využívanými vozidlami $s_1, s_2, \cdots, s_m$ dispečeri budú proti tomu protestovať. Cieľom je nájsť také permutácie denných dodávok v stĺpcoch matice A, pri ktorých sú celkové dodávky čo najrovnomernejšie.

Bez straty všeobecnosti môžeme predpokladať, že permutácia prvého stĺpca matice je identická, t. j. $\psi_1 = (1, 2, \ldots, m)$. Zvláštnu úlohu to totiž zohráva v špeciálnom prípade – **probléme permutácie dvojstĺpcovej matice** MPP2. Platí

$$F_{sqr}(s_1, \ldots, s_m) = \sum_{i \in M} \left(a_{i1} + a_{\psi_{i2},2} - \overline{s}\right)^2 = \sum_{i \in M} \left(a_{i1} + a_{\psi_{i2},2}\right)^2 +$$

$$2\overline{s} \sum_{i \in M} a_{i1} + 2\overline{s} \sum_{i \in M} a_{i2} + m\overline{s}^2 = \sum_{i \in M} \left(a_{i1} + a_{\psi_{i2},2}\right)^2 + 4\overline{s}^2 + m\overline{s}^2.$$

[14] Nech $I \subset \mathbb{R}$ je interval reálnych čísel. Funkciu $I^n \to \mathbb{R}$ nazývame Shur-konvexnou, ak pre ľubovoľnú dvojstochastickú maticu $\mathrm{S} \in \mathbb{R}^{m \times m}$ (nezáporná s jednotkovými riadkovými a stĺpcovými súčtami) platí $F(\mathbf{x}\mathrm{S}) \leq F(\mathbf{x})\ \forall \mathbf{x} \in I^n$.

A tak MPP2 môžeme formulovať ako nasledujúcu priraďovaciu úlohu:

$$z = \sum_{i\in M}\sum_{j\in M}\left(a_{i1}+a_{j2}\right)^2\cdot x_{ij} \to \min \tag{4.83}$$

$$\sum_{j\in M} x_{ij} = 1 \qquad i\in M, \tag{4.84}$$

$$\sum_{i\in M} x_{ij} = 1 \qquad j\in M, \tag{4.85}$$

$$x_{ij}\in\{0,1\} \qquad i\in M, j\in M, \tag{4.86}$$

kde položíme $\psi_{i2} = j$, keď $x_{ij} = 1$. Tým sme ale ukázali, že MPP je riešiteľný ako priraďovací problém v polynomiálnom čase $O(m^3)$.

Možno ale ukázať viac. Ak stĺpce dvojstĺpcovej matice $\mathrm{A} = (a_{ij}) \in \mathbb{R}^{m\times 2}$ usporiadame do matice

$$\mathrm{A}^{\psi} = \begin{pmatrix} a_{\psi_{11},1} & a_{\psi_{12},2} \\ a_{\psi_{21},1} & a_{\psi_{22},2} \\ \vdots & \vdots \\ a_{\psi_{m1},1} & a_{\psi_{m2},2} \end{pmatrix}$$

tak, že

$$a_{\psi_{11},1} \le a_{\psi_{21},1} \le \cdots \le a_{\psi_{m1},1},$$
$$a_{\psi_{12},2} \ge a_{\psi_{22},2} \ge \cdots \ge a_{\psi_{m2},2},$$

potom $\psi = (\psi_1, \psi_2)$ je optimálnym riešením MPP2.

Nakoľko usporiadanie m čísel možno realizovať algoritmom *quicksort*, ktorý má zložitosť $O\big(m\cdot log_2(m)\big)$, dostávame $O\big(m\cdot log_2(m)\big)$ exaktný algoritmus pre MPP2, ktorý zotriedi prvý stĺpec matice vzostupne a druhý stĺpec zostupne.

Vzhľadom na algoritmickú zložitosť MPP vo všeobecnom prípade do úvahy prichádzajú heuristické metódy riešenia. Najúspešnejšou sa ukázala nasledujúca stochastická *metóda dvojstĺpcovej kumulácie matice* založená na opakovanom riešení problému MPP2 na náhodne kumulovaných dvoriadkových maticiach:

Krok 1: Štartujeme s $m \times n$ maticou A a ľubovoľnou n-ticou $\psi = (\psi_1, \psi_2, \ldots, \psi_n)$ permutácií množiny M.

Krok 2: Náhodne vyberieme neprázdnu podmnožinu $J \neq \emptyset$, $J \subset N$ a definujeme množinu $K = N - J$.

Krok 3: Riešime MMP2 s riešením $\pi = (\pi_1, \pi_2)$ a $m \times 2$ permutovanou maticou $\mathrm{B}^\pi = (b^\pi_{ij})$ kde

$$b^\pi_{i1} = \sum_{j \in J} a_{\pi_{ij},j}, \quad b^\pi_{i2} = \sum_{j \in K} a_{\pi_{ij},j}$$

usporiadaním jej prvého riadku vzostupne a druhého stĺpca zostupne s optimálnou dvojicou permutácií $\pi = (\pi_1, \pi_2)$.

Krok 4: Aplikujeme permutácie π_1 na všetky permutácie ψ_{j1} pre $j \in J$ a podobne π_2 na všetky permutácie ψ_{j2} pre $j \in K$, t. j. obnovíme ψ takto:

$$\psi_j = \begin{cases} \pi_1 \circ \psi_j & \text{ak } j \in J \\ \pi_2 \circ \psi_j & \text{ak } j \in K \end{cases}$$

Krok 5: Ak nejaké zastavovacie kritérium je splnené, potom **STOP** s riešením ψ. Ináč **GOTO KROK 2**.

Metóda dvojstĺpcovej kumulácie matice využíva skutočnosť, že každá kumulácia stĺpcov matice A do dvojriadkovej matice A v **Kroku 3** je invariantom, a teda aj nutnou podmienkou optimality MPP. Žiaľ možno dokázať, že nie je podmienkou postačujúcou.

Uvažujme nasledujúci **vážený model rovnomerného zásobovania skladov**. Nech sú splnené predpoklady základného modelu rovnomerného zásobovania skladov. Naviac pre každý i-ty sklad sú známe skladovacie náklady w_i na jednotku komodity. Celkové skladovacie náklady i-teho skladu vytvorené dodávkami vozidiel sú $s_i = w_i \sum_{j=1}^{n} a_{ij}$. Ak existujú veľké rozdiely medzi takto vytvorenými skladovacími nákladmi v skladoch $s_1, s_2, \cdots, s_m$, skladníci budú opäť nespokojní. Cieľom je teda nájsť také permutácie dodávok v stĺpcoch matice A, pri ktorých sú kumulované skladovacie náklady v skladoch čo najrovnomernejšie.

Problémom permutácií váženej matice WMPP (weighted matrix permutation problem [8]) rozumieme nasledujúcu úlohu: Nech sú $M = \{1, 2, \ldots, m\}$ a $N = \{1, 2, \ldots, n\}$ konečné neprázdne množiny indexov danej reálnej matice $\mathrm{A} = (a_{ij}) \in \mathbb{R}^{m \times n}$ a daná n-tica kladných reálnych váh $(w_1, w_2, \ldots, w_n) \in \mathbb{R}^n_+$. Hľadá sa sa taká usporiadaná n-tica $\psi = (\psi_1, \psi_2, \ldots, \psi_n)$ permutácií stĺpcov M matice A, pre ktorú je

$$F(w_1 s_1, w_2 s_2, \ldots, w_m s_m) \to \min \tag{4.87}$$

$$s_i = \sum_{j \in N} a_{\psi_{ij},j} \qquad i \in M, \tag{4.88}$$

$$\psi_j = (\psi_{1j}, \psi_{2j}, \ldots, \psi_{mj}) \in \Pi(N) \qquad j \in N, \tag{4.89}$$

kde F je nejaká Shur-konvexná funkcia.

Problém MPP je vlastne prípadom WMPP s konštantnými (jednotkovými) váhami riadkov matice, čo sa prejaví v argumentoch cieľovej funkcie (4.87). Vo funkcii mier nerovnomernosti môžeme opäť použiť funkcie (4.80) a (4.81). V prípade miery analogickej (4.82) však musíme ináč definovať „ideálny vážený riadkový súčet“, označme ho $\tilde{s_w}$. Nakoľko pre optimálne riešenie ψ je

$$\sum_{j \in N} a_{\psi_{ij},j} \approx \frac{\tilde{s_w}}{w_i} \qquad \forall i \in M \ . \tag{4.90}$$

Súčet pravých strán (4.90) je ale súčtom prvkov matice A, a tak máme kvadratickú mieru nerovnomernosti v tvare

$$F_{sqr}(w_1 s_1, w_2 s_2, \ldots, w_n s_m) = \sum_{i \in M} (w_i s_i - \overline{s_w})^2, \ \tilde{s_w} = \frac{\sum_{i \in M} \sum_{j \in N} a_{ij}}{\sum_{i \in M} 1/w_i} \ ,$$

čo je pre jednotkové váhy v zhode s (4.82).

Na rozdiel od MPP v úlohe WMPP už nemôžeme bez straty všeobecnosti fixovať prvý stĺpec matice, čo nám znemožňuje formulovať špeciálny prípad – **problém permutácie váženej dvojstĺpcovej matice** WMPP2 ako priraďovací problém. Môžeme ho ale na základe vzťahu (4.90) transformovať na trojstĺpcový MPP – MPP3 s maticou B_A

$$\mathrm{B}_A = \begin{pmatrix} a_{11} & a_{12} & -b_1 \\ a_{21} & a_{22} & -b_2 \\ \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & -b_m \end{pmatrix} \tag{4.91}$$

kde $b_i = \dfrac{\tilde{s_w}}{w_1}$ je i-ty „ideálny doplnkový riadkový súčet“. Poznamenajme, že v tejto úlohe je cieľom nulová hodnota kriteriálnej funkcie.

Problém WMPP2 s maticou $\mathrm{A} \in \mathbb{R}^{m\times 2}$ môžeme riešiť ako úlohu MMP3 s maticou B_A určenou (4.91) pomocou už uvedenej metódy dvojstĺpcovej kumulácie matice. Táto metóda je využiteľná aj vo všeobecnom prípade matice $\mathrm{A} \in \mathbb{R}^{m\times n}$. Stačí totiž len nahradiť v **Kroku 3** metódu riešenie problému MPP2 metódou riešenia problému WMPP2.

4.7 Metódy preberania tovaru

Počas pohybu materiálu, substrátov či výrobkov v logistickom reťazci nastávajú situácie, kedy treba rozhodnúť, či kvalita spomínaných entít zodpovedá požiadavkám, a v tom prípade pokračovať v ich postupe logistickým reťazcom, alebo ich v prípade nedostatočnej kvality entity vrátiť v logistickom reťazci o krok späť. Najčastejšie takáto situácia nastáva pri zmene majiteľa tovaru, ale potreba rozhodovania o prijatí či odmietnutí dodávky môže vzniknúť i pri prechode článkami logistického reťazca u jedného majiteľa – napríklad pri prechode medzi etapami výrobného procesu. V tejto časti sa budeme zaoberať preberacími procedúrami diskrétnych výrobkov.

V rámci dnes rozšíreného prístupu TQM (total quality management) sa preferuje 100% kontrola výrobkov. Tá však nie je vždy možná z ekonomických alebo i principiálnych dôvodov. Stopercentná kontrola môže byť totiž príliš drahá, alebo test kvality môže deštruktívny (napríklad testy hraničnej pevnosti, testy konzerv, zápaliek a pod.).

V spomínaných prípadoch sa musíme uchýliť k štatistickým testom. Základným princípom štatistického preberania je rozhodovanie o prijatí či odmietnutí celej dávky výrobkov na základe vyhodnotenia náhodne vybratej vzorky. Označme dávku s N výrobkami, z ktorých je D defektných, symbolom $\mathcal{D}(N, D)$. Dávka sa pokladá za dobrú, keď obsahuje najviac $p * 100\%$ chybných výrobkov – t. j. ak $D \leq pN$.

Stanoviť *vzorkovací plán* znamená pre dávku veľkosti N určiť veľkosť vzorky n a číslo c – maximálny počet defektných výrobkov vo vzorke, pre ktoré ešte dávku pokladáme za dobrú. Takýto vzorkovací plán označme $\mathcal{S}(n, c)$

Predpokladajme, že výrobky prichádzajú v dávkach obsahujúcich N výrobkov. Skúmajme dávku $\mathcal{D}(N, D)$. Vyberme náhodne n výrobkov z dávky. Aká je pravdepodobnosť, že medzi n náhodne vybratými výrobkami bude práve d defektných?

Počet všetkých možností, ako vybrať z N prvkov n-prvkovú podmnožinu, je $\binom{N}{n}$, z nich práve $\binom{D}{d}.\binom{N-D}{n-d}$ obsahuje d chybných výrobkov. Preto náhodná

veličina X definovaná ako počet chybných výrobkov v n náhodne vybratých výrobkov má rozdelenie dané vzťahom

$$P(X = d) = P(X = d, N, D, n) = \frac{\binom{D}{d}.\binom{N-D}{n-d}}{\binom{N}{n}} .$$

Takéto rozdelenie sa nazýva *hypergeometrické rozdelenie*. Jeho stredná hodnota je

$$\mu = \frac{n.D}{N}$$

a rozptyl

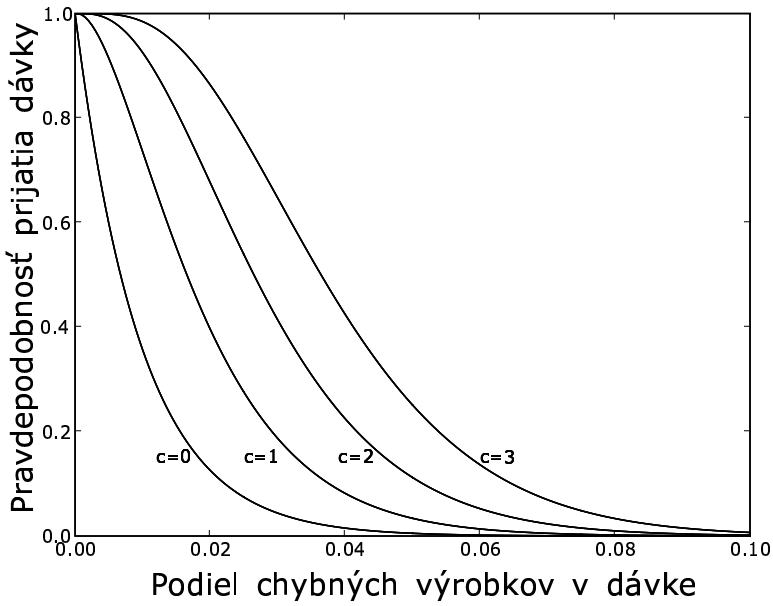
$$\sigma^2 = \left(\frac{nD}{N}\right)\left(1 - \frac{D}{N}\right)\left(\frac{N-n}{N-1}\right) .$$

Majme dávku s veľkosťou N a vzorkovací plán $\mathcal{S}(n, c)$. S takýmto vzorkovacím plánom sú spojené dve riziká

- *Riziko dodávateľa.* Toto riziko je rovné pravdepodobnosti, že vzorkovací plán zamietne dávku napriek tomu, že neobsahuje viac ako $p * 100\%$ chybných výrobkov.
- *Riziko odberateľa.* Toto riziko je rovné pravdepodobnosti, že vzorkovací plán prijme dávku napriek tomu, že obsahuje viac ako $p * 100\%$ chybných výrobkov.

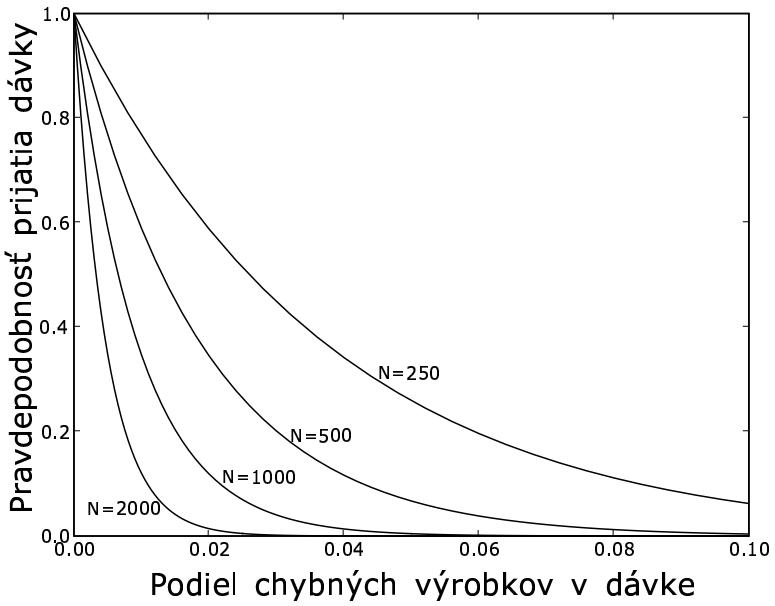
Pravdepodobnosť prijatia dávky $\mathcal{D}(N, D)$ vzorkovacím plánom $\mathcal{S}(n, c)$ je

$$\begin{aligned} F_X(c) = F_X(c, N, D, n) = P(X \leq c) = \\ = P(X = 0, N, D, n) + P(X = 1, N, D, n) + \cdots + P(X = c, N, D, n) = \\ = \frac{\binom{D}{0}.\binom{N-D}{n-0}}{\binom{N}{n}} + \frac{\binom{D}{1}.\binom{N-D}{n-1}}{\binom{N}{n}} + \cdots + \frac{\binom{D}{c}.\binom{N-D}{n-c}}{\binom{N}{n}} . \end{aligned}$$



Obr. 4.13: Závislosti pravdepodobnosti prijatia dávky veľkosti 2000 od podielu chybných výrobkov pre rôzne vzorkovacie plány s pevnou veľkosťou vzorky $n = 100$ ($N = 2000$, $n = 100$, $c = 0, 1, 2, 3$)

Závislosť prijatia dávky $\mathcal{D}(N, D)$, kde $D = p.N$, od podielu p chybných výrobkov pri rôznych vzorkovacích plánoch $\mathcal{S}(n, c)$ s pevnou veľkosťou vzorky n vidíme na obrázku 4.13. So zvyšovaním povoleného počtu c chybných výrobkov vo vzorke stúpa pravdepodobnosť prijatia dávky, čo je všeobecne očakávaná skutočnosť.

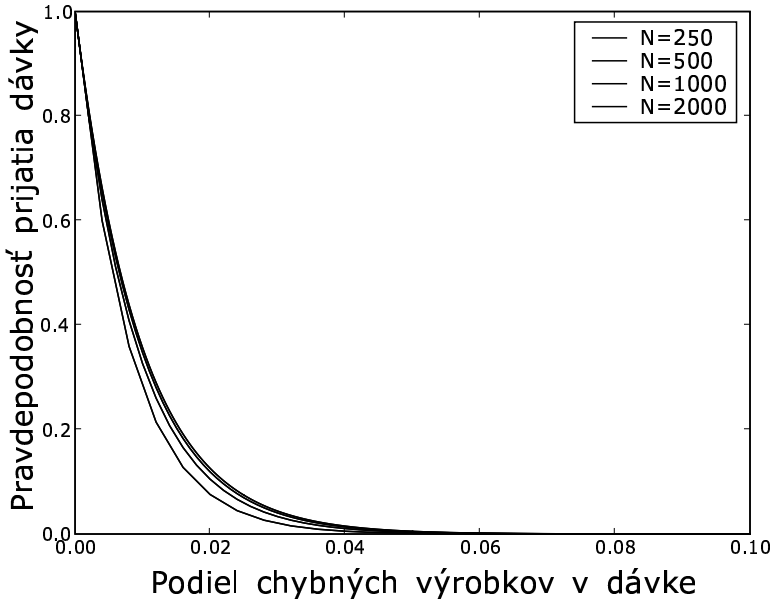


Obr. 4.14: Závislosti pravdepodobnosti prijatia dávok rôznej veľkosti od podielu chybných výrobkov pre vzorkovacie plány s jednou desatinou testovaných výrobkov – $n = N/10$ $(c = 0,\ N = 250,\ 500,\ 1000, 2000)$

Menej očakávanú skutočnosť vyjadruje graf na obrázku 4.14. Pravdepodobnosť prijatia dávky pri vzorkovaní 10% celej dávky výrazne klesá s veľkosťou dávky.

Graf na obrázku 4.15 ukazuje závislosť pravdepodobnosti prijatia dávky s podielom p chybných výrobkov pri vzorkovacom pláne $\mathcal{S}(100, 0)$ pre veľkosti dávky pre rôzne hodnoty veľkosti dávky N. Ukazuje sa, že pravdepodobnosti prijatia dávky málo závisia od jej veľkosti.

Z uvedenej skutočnosti vyplýva, že pri navrhovaní preberacieho plánu a stanovovaní veľkosti dávky treba maximalizovať veľkosť dávky.



Obr. 4.15: Závislosti pravdepodobnosti prijatia dávok rôznej veľkosti od podielu chybných výrobkov pre vzorkovací plán s veľkosťou testovacej vzorky $n = 100$ ($c = 0$, $N = 250,\ 500,\ 1000, 2000$)

4.8 Bayesovské rozhodovanie o veľkosti objednávky

4.8.1 Princíp bayesovského rozhodovania

Nech $S = \{s_1, s_2, \ldots, s_m\}$ je množina očakávaných všetkých situácií, ktoré môžu nastať. Nech pravdepodobnosť toho, že nastane situácia s_i, je p_i. Je zrejme $\sum_{i=1}^{m} p_i = 1$. Nech $T = \{t_1, t_2, \ldots, t_n\}$ je množina stratégií, ktoré môžeme voliť. Ak nastane situácia s_i a my sme zvolili stratégiu t_j, dostaneme zisk z_{ij}.

Sme v situácii, kedy sa musíme dopredu rozhodnúť pre nejakú zo stratégií z T, keď nevieme, ktorá zo situácií z množiny S nastane.

Ak zvolíme stratégiu t_j, dostaneme stredný zisk

$$Z(j) = \sum_{i=1}^{m} z_{ij}.p_i \ . \tag{4.92}$$

Z hľadiska dlhodobého rozhodovania sa treba rozhodnúť pre tú stratégiu, ktorá maximalizuje strednú hodnotu zisku.

Praktickým problémom tu môže byť určenie pravdepodobností p_i. Jedným zo spôsobov je využitie historických údajov o početnosti výskytu situácií s_i pre $i = 1, 2, \ldots, m$ a na základe týchto početností odhadnúť pravdepodobnosť p_i ako podiel početnosti výskytu situácie s_i k počtu výskytu všetkých situácií. Takéto pravdepodobnosti sa niekedy volajú *aposteriórne pravdepodobnosti*.

Pokiaľ historické údaje nie sú k dispozícii, alebo keď sa podmienky pre výskyt uvažovaných situácie diametrálne zmenili, možno pravdepodobnosti situácií p_i odhadnúť na základe nejakých skutočností súvisiacich s uvažovanými situáciami. Takéto pravdepodobnosti nazývame *apriórne pravdepodobnosti*.

4.8.2 Rozhodovanie o veľkosti objednávky za neurčitosti

V tejto časti budeme riešiť nasledujúcu úlohu: Obchodník predáva výrobky, po ktorých je náhodný dopyt. Problémom je, koľko výrobkov treba objednať, aby mal obchodník čo najvyšší zisk. Ak objedná viac ako predá, zisk z predaja mu zníži strata z nepredaných výrobkov. Ak objedná menej, ako zákazníci požadujú, nevyužije príležitosť zarobiť predajom viac. Na riešenie tohto problému dáva nástroje bayesovská rozhodovacia teória.

Označme s_i pre $i = 0, 1, 2, \ldots, n$ tú situáciu, kedy bude dopyt po i výrobkoch. Nech p_i je pravdepodobnosť, že nastane situácia s_i, t. j. že obchodník za predpokladu, že má k dispozícii dostatočný počet výrobkov, predá práve i výrobkov. Označme t_j stratégiu, kedy obchodník objedná presne j výrobkov, $j = 0, 1, 2, \ldots, m$. Je nezmysel objednávať viac výrobkov, než je šanca predať, preto musí byť $m \leq n$. V tomto štádiu však niet dôvodu, prečo by sme mali vylúčiť stratégiu t_n, preto položme $n = m$.

Označme z_{ij} zisk obchodníka v prípade, keď objednal j výrobkov a dopyt bol i výrobkov. Obchodník musí objednávať – t. j. zvoliť niektorú stratégiu t_j – dopredu, pričom nevie, aký bude dopyt. Vie však, že pri voľbe stratégie t_j bude jeho stredný zisk $Z(j)$ daný vzťahom 4.92 na strane 173, a teda z dlhodobého hľadiska bude optimálna tá stratégia t_j, ktorá maximalizuje $Z(j)$.

V najjednoduchšom prípade môžeme maticu ziskov $\mathbf{Z} = \{z_{ij}\}$ určiť takto: Označme c zisk z predaja jedného výrobku a d stratu, ktorú má obchodník, ak

sa mu objednaný a dodaný výrobok nepodarí predať. Potom

$$z_{ij} = \begin{cases} c.i - d.(j-i) & \text{ak } i \leq j \\ c.j & \text{ak } j < i \end{cases} \tag{4.93}$$

Stredná hodnota zisku pri stratégii j je

$$\begin{aligned} Z(j) &= \sum_{i=1}^{n} z_{ij} p_i = \sum_{i;i\leq j} z_{ij} p_i + \sum_{i;i>j} z_{ij} p_i = \\ &= \sum_{i;i\leq j} [c.i - d.(j-i)].p_i + \sum_{i;i>j} c.j.p_i = \\ &= (c+d). \sum_{i;i\leq j} i.p_i - d.j. \sum_{i;i\leq j} p_i + c.j. \sum_{i;i>j} p_i \ . \end{aligned} \tag{4.94}$$

Označme

$$F(j) = \sum_{i;i\leq j} p_i \ . \tag{4.95}$$

Funkcia $F(j)$ vyjadruje pravdepodobnosť, že dopyt bude menší alebo rovný ako j. Je to vlastne distribučná funkcia náhodnej veličiny veľkosti dopytu. Potom

$$\sum_{i;i>j} p_i = 1 - \sum_{i;i\leq j} p_i = 1 - F(j) \ . \tag{4.96}$$

Dosadením z (4.95) a (4.96) do (4.94)

$$\begin{aligned} Z(j) &= (c+d). \sum_{i;i\leq j} i.p_i - d.j.F(j) + c.j.[1 - j.F(j)] = \\ &= (c+d). \sum_{i;i\leq j} i.p_i - (c+d).j.F(j) + c.j = \\ &= (c+d).\Big[\sum_{i;i\leq j} i.p_i - j.F(j) \Big] + c.j \ . \end{aligned}$$

Máme teda

$$Z(j) = (c+d).\Big[\sum_{i;i\leq j} i.p_i - j.F(j) \Big] + c.j \ . \tag{4.97}$$

Počítajme prírastok zisku $Z(j+1) - Z(j)$.

$$\begin{aligned} Z(j+1) \quad - \quad & Z(j) = \\ & (c+d).\Big[\sum_{i;i\le j+1} i.p_i - (j+1).F(j+1)\Big] + c.(j+1) - \\ & \qquad -(c+d).\Big[\sum_{i;i\le j} i.p_i - j.F(j)\Big] + c.j = \\ = \quad & (c+d).\Big[(j+1).p_{j+1} - (j+1).(F(j)+p_{j+1}) + j.F(j)\Big] + c = \\ = \quad & (c+d).\Big[-(j+1).F(j) + j.F(j)\Big] + c = c - (c+d).F(j)\ . \end{aligned}$$

Máme teda

$$Z(j+1) - Z(j) = c - (c+d).F(j)\ . \tag{4.98}$$

Funkcia F je neklesajúca funkcia, preto prírastok zisku (4.98) je nerastúca funkcia premennej j. Preto s rastúcim počtom j objednaných výrobkov rastie aj celkový zisk dovtedy, kým je prírastok zisku $Z(j+1) - Z(j)$ kladný.

Hľadajme hraničnú hodnotu funkcie $F(j)$, pri ktorej sa mení prírastok (4.98) z kladného na záporný. Zrejme táto hodnota hodnotu spĺňa rovnicu

$$c - (c+d).F(j) = 0\ ,$$

odkiaľ

$$F(j) = \frac{c}{c+d}\ . \tag{4.99}$$

Ak by teda pre nejaké j platilo (4.99), toto j by maximalizovalo zisk $Z(j)$ daný vzťahom (4.97).

Pretože však definičný obor funkcie $F(j)$ je diskrétna množina, ($F(j)$ je definované pre j=0,1,2,. . . ,n), nemusí existovať také j, pre ktoré by platilo (4.99). V tom prípade maximum zisku nájdeme nasledujúco. Označme j_0 také celé číslo, pre ktoré platí

$$F(j_0) < \frac{c}{c+d} < F(j_0+1)\ .$$

Potom väčšie z čísel $Z(j_0)$, $Z(j_0+1)$ je maximum funkcie $Z(j)$ danej vzťahom (4.97).

4.8.3 Pravdepodobnosť straty

Nech s_{ij} je strata, ak pri stratégii t_j – t. j. pri objednaných j výrobkoch predá obchodník i výrobkov.

$$s_{ij} = \begin{cases} d.(j-i) - c.i & \text{ak } i \leq j \\ 0 & \text{ak } j < i \end{cases} \tag{4.100}$$

Nech $S(j)$ je náhodná veličina predstavujúca stratu pri stratégii j.
Platí $S(j) < s$ práve vtedy, keď $d.(j-i) - c.i < s$, odkiaľ $-(c+d).i < -d.j + s$, a teda $i > \dfrac{d.j - s}{c+d}$.
Počítajme pravdepodobnosť $P(S(j) < s)$, že strata pri stratégii j bude menšia než hodnota s.

$$\begin{aligned} P(S(j) < s) &= P\left(i > \frac{d.j - s}{c+d}\right) = \\ &= 1 - P\left(i \leq \frac{d.j - s}{c+d}\right) = 1 - F\left(\left[\frac{d.j - s}{c+d}\right]\right), \end{aligned}$$

kde hranatá zátvorka $[x]$ predstavuje celú časť reálneho čísla x. Z posledného vzťahu s využitím vlastnosti $P(S(j) \geq s) = 1 - P(S(j) < s)$ dostávame

$$P(S(j) \geq s) = F\left(\left[\frac{d.j - s}{c+d}\right]\right) . \tag{4.101}$$

Pri pevnej hodnote s s rastúcim j rastie aj pravdepodobnosť $P(S(j) \geq s)$ – javu, že strata prekročí hodnotu s.

Pri bayesovskom rozhodovaní o veľkosti objednávky je vhodné uvážiť aj pravdepodobnosť takého javu, že strata prekročí hodnotu, ktorá by mohla objednávateľa poškodiť. Ak by táto pravdepodobnosť bolo príliš vysoká, možno znížiť objednávané množstvo. Zníži sa tak nebezpečenstvo úpadku objednávateľa za cenu zníženia jeho stredného očakávaného zisku.

Register

Literatúra

[1] AHR, S.: *A Survey of Arc Routing Problems*, prístupné na `http://www.informatik.uni-heidelberg.de/groups/comopt/people/ahr/research/overviewArcRouting/`, (2003)

[2] BERKELAAR, M., EIKLAND, K., NOTEBAERT, P.: *lp_solve - Open source (Mixed-Integer) Linear Programming system*, prístupné na `http://groups.yahoo.com/group/lp_solve`

[3] BREZINA, I., ČIČKOVÁ, Z., REIFF, M.: *Kvantitatívne metódy v logistike, Zbierka príkladov*, Vydavateľstvo Ekonóm, Bratislava, (2005), ISBN 80–225–1967–7

[4] BREZINA, I., IVANIČOVÁ, Z.: *Kvantitatívne metódy v logistike*, Vydavateľstvo Ekonóm, Bratislava, (1999), ISBN 80–225–1735–6

[5] CONWAY, W.,R., MAXWELL, W.,L.,MILLER, L.,W.: *Theory of Scheduling.* Addison-Wesley Publishing Company, (1967)

[6] ČERNÝ, J.: *Multi–Polarized Graphs.* Apl.Mat. 21, str. 145–147, (1976)

[7] ČERNÝ, J.: *Fleet management – selected optimization problems*, Proc. 8th IFAC-IFIF-IFORS Symp. „Transportation systems“, Cania (Greke) June, 607-610, (1997),

[8] ČERNÝ, J.,PEŠKO, Š.: *A Note to Weighted Matrix Permutation Problem*, `http://frcatel.fri.utc.sk/~pesko/ipubl.html`, (2004)

[9] DEMEL, J.: *Grafy*, SNTL, Praha (1989)

[10] DONALDSON, W. A.: *Inventory Replenishment Policy for a Linear Trend in Demand - an Analytical Solution*,J. Opl. Res. Soc. 28, str. 663-670, (1977)

[11] EVANS, J.R., MINIEKA, E.: *Optimization Algorithms for Networks and Graphs*, Marcel Dekker, Inc. (1992), second edition, ISBN 0–8247–8602–5

[12] GÁBRIŠOVÁ, L.: *Limits of the Multiplier Adjustment Approach to Capacitated Location Problem*, In: Scietific Letters of the University of Žilina, Communications 4/2005, str. 52-55, (2005), ISSN 1335–4205

[13] GÁBRIŠOVÁ, L.: *Combination of the Methods Based on Lagrangean Relaxation of the Capacity Constraints in the Facility Location Problem*, In: Proceedings of the Challenges in Transport and Communication, 14-15 September 2006, Pardubice, str. 953-958, (2006), ISBN 80–7194–880–2

[14] GNAP, J.: *Možnosti využitia programu TRASMAP vo vzdelávaní v odbore cestnej dopravy*, MEDACTA 97, č.4, str.1108-1111, (1997), ISBN 80–967339–9–0

[15] GROSS, J., YELLEN, J.: *Graph Theory and its Applications*, CRC Press (1999), ISBN 0–8493–3982–0

[16] HILL, R.M.: *Inventory Model for Increasing Demand Followed by level Demand*, J. Opl. Res. Soc. 46, str. 1250-1259, (1995)

[17] HINDI, K. S.: *Eficient Solution of Single-item, Capacited Lot-sizing Problem with Start-ut and Reservation Cost* J. Opl. Res. Soc. 46, str. 1223-1236, (1995)

[18] HUŠEK, R., MAŇAS, M.: *Matematické modely v ekonomii*, SNTL Praha, (1989), ISBN 80–03–00098–X

[19] CHRISTOFIDES, N.: *Graph theory – an Algorithmic Approach*, Academic Press, London (1975). Ruský preklad, Moskva (1978)

[20] CHOBOT, M., TURNOVCOVÁ, A.: *Modely rozhodovania v konfliktných situáciách a za neurčitosti* ALFA Bratislava, (1980), 63–066–80

[21] JÁNOŠÍKOVÁ, Ľ., SADLOŇ, Ľ., CENEK, J.: *Model of Inteligent Transportation Infrastructure*, Journal of Information, Control and Management Systems, Faculty of Management Science and Informatics, University of Žilina, Vol.1, No.1., pp.47-56, (2003), ISSN 1336–1716

[22] JANÁČEK, J.: *Matematické programování*, Žilinská univerzita v Žiline, (1999), ISBN 80–71000–573–8

[23] JANÁČEK, J.: *Optimalizace v dopravních sítích*, Žilinská univerzita v Žiline, (2003), ISBN 80–8070–031–1

[24] JANÁČEK, J., JANOŠÍKOVÁ, Ľ.: *Směrová lokace uzlových stanic - nástroj k omezení protisměrných přeprav*, Zborník prednášok 7. sympózia ŽEL 2000, Žilina, 30-31.máj, EDIS – vydavateľstvo ŽU, str.115-119, (2000)

[25] JANÁČEK, J., JANÁČKOVÁ, M.: *Sekvenční algoritmy řešení úlohy okružních jízd*, Horizonty dopravy 1, str.44-53, (1992)

[26] JANÁČEK, J., GÁBRIŠOVÁ, L.: *Fuzzy Set Operations for Capacity Contraint Relaxation*, In: Journal of Information, Control and Management Systems, Vol.3, No.2, str. 81-89, (2005), ISSN 1336–1716

[27] JANÁČEK, J., GÁBRIŠOVÁ, L.: *Lagrangean Relaxation Based Approximate Approach to the Capacitated Location Problem.Lagrangean Relaxation Based Approximate Approach to the Capacitated Location Problem*, In: Scietific Letters of the University of Žilina, Communications 3/2006, str. 19-24, ISSN 1335–4205

[28] KRÁL, J.: *Podniková logistika - Riadenie dodávateľského reťazca*, EDIS – vydavateľstvo ŽU, Žilina, (2001), ISBN 80–7100–864–8

[29] KUČERA, L.: *Kombinatorické algoritmy*, Matematický seminář SNTL, Praha (1983)

[30] LAWLER, E. L., LENSTRA, J. K., KAN, A.H.G.R., SMOYS, D. B.: *The traveling salesman Problem*, JOHN WILEY & SONS, (1985)

[31] LI, S., X., HUANG, Z., ASLEY, A.: *Seller-buyer System Co-operation in a Monopolistic Market* J. Opl. Res. Soc. 46, str. 1456-1470, (1995)

[32] LINDA, B.: *Stochastocké modely operčního výzkumu*, Statis, Bratislava (2004), ISBN 80–8569–33–6

[33] MIČIETA, B., KRÁL, J.: *Plánovanie a riadenie výroby*, ŽU Žilina, (1998) ISBN 80–7100–430–8

[34] MESSINA, W., S.: *Statistical Quality Control for Manufacturing Managers*, J. Willey & Sons,(1987), ISBN 0–471–85774–2

[35] NEŠETŘIL, J.: *Teorie grafů*, Matematický seminář, SNTL, Praha (1979)

[36] PALÚCH, S.: *Optimalizácia zvozov a rozvozov*, Zborník prednášok z vedeckého seminára Doprava a manipulácia s poľnohospodárskymi materálmi, str. 88-94, (1998)

[37] PEŠKO, Š.: *Optimalizácia rozvozov s časovými oknami* konferencia KYBERNETIKA - história, perspektívy, teória a prax, zborník referátov, Žilina, (2002), ISBN 80–967609–7–1

[38] PEŠKO, Š.: *Pyramidová metóda pre úlohu obchodného cestujúceho*, Komunikácie 4, Vedecké listy Žilinskej univerzity, str. 29-34, (2000), ISSN 1335–4205

[39] PEŠKO, Š.: *Optimalizácia NP-ťažkých dopravných rozvrhov*, Habilitačná práca, ŽU Žilina, str. 120, (2002)

[40] PEŠKO, Š.: *The Matrix Permutation Problem in Interval Arithmetic*, In: Journal of Information, Control and Management Systems, Vol.4, No.1, pp 35-40, (2006), ISSN 1336–1716

[41] PEŠKO, Š., ČERNÝ, J.: *Uniform Splitting in Managerial Decision Making*, Economics and Management, Vol. 4, pp. 67-72, (2006), ISSN 1212–3609

[42] PINEDO, M.: *Scheduling: Theory, Algorithms, and Systems (2nd Edition)*, Prentice Hall; 2 edition (August 8, 2001), ISBN: 0130281387

[43] PLESNÍK, J.: *Grafové algoritmy*, Veda, Bratislava (1983)

[44] PONČÁK, M.: *The fastidious Travelling Problem*, Transcom 2003, University of Žilina. 5-th European Conference of Young Research and Science Workers in Transport and Telecommunication, (2003), ISBN 80–8070–079–6

[45] ROSEN, K.H.: *Handbook of Discrete and Combinatorial Mathematics*, CRC Press (2000), ISBN 0–8493–0149–1

[46] ROSLING, K.: *The (r,Q) Inventory Model with Lost Sales*, School of Industrial Engineering Växjo University, Report SE-39195, (2002)

[47] SAFRA, S., SCHWARTZ, O.: *On the complexity of approximating TSP with neighborhoodds and related problems*, In Proc. ESA, str. 448-458, (2003)

[48] SEDLÁČEK, J.: *Úvod do teorie grafů*, Academia Praha, (1981)

[49] STANKOVIANSKA, I.: *Hranovokapacitné úlohy na dopravných sieťach*, dizertačná práca, Žilina, (1996)

[50] STANKOVIANSKA, I.: *A Comparision of Fletcher-Clark and Golden-Wong Algorithm*, Práce a štúdie Fakulty riadenia VŠDS, Žilina, (1992)

[51] STANKOVIANSKA, I.: *An Effective Approach to Routing Problems*, Scientific Papers of the University of Pardubice - Series D, Faculty of Economics and Administratin 10, str.148-151, (2006), ISBN 80–7194–851–9

[52] TAGZE, M., VLACH, M.: *The matrix permutation Problem*, Report 84-54, Tech. univ. Graz, (1984)

[53] VAŠEK, K., PEŠKO, Š.: *Dopravné rozvrhy a ich optimalizácia*, Výskumná správa III-8-6/09.3, Výskumný ústav dopravný, Žilina, (1983)

[54] ZHANG, X. M.: *Optimization of Schur Convex Function*, Mathematical Inequalities and Applications, Vol 1, No.3, 319-323 (1998)

[55] ZELINKA, B.: *Polar Graphs and railway traffic*, Aplikace Mat. 19, 169-176, (1974)

doc. RNDr. Stanislav Palúch, CSc., doc. RNDr. Štefan Peško, CSc.

KVANTITATÍVNE METÓDY V LOGISTIKE

Žilinská univerzita v Žiline, Univerzitná 8215/1 010 26 Žilina
v edičnom rade VEDECKÉ MONOGRAFIE
Vedecký redaktor Prof. Ing. Mikuláš Alexík, CSc.
Zodpovedný redaktor PhDr. Katarína Šimánková
Technický redaktor Mária Závodská

Vytlačilo EDIS – vydavateľstvo ŽU, J. M. Hurbana 15, Žilina
v decembri 2006 ako svoju 2328. publikáciu.
185 strán, 52 obrázkov, 2 tabuľky, AH 11,22, VH 11,46
prvé vydanie, náklad 200 výtlačkov
Typografický systém LaTeX pod o. s. LINUX
Grafická úprava obálky Juraj Zbýňovec
ISBN 80-8070-636-0