Technická univerzita
v Košiciach

Ján Paralič

ROZVRHOVANIE A LOGISTIKA

Košice

doc. Ing. Ján Paralič, PhD.
Katedra kybernetiky a umelej inteligencie
Fakulta elektrotechniky a informatiky
Technická univerzita v Košiciach
Jan.Paralic@tuke.sk

Lektoroval: Ing. Peter Butka

ISBN

Obsah

Predhovor

Predkladaný učebný text je súhrnom podkladov k prednáškam z predmetu Rozvrhovanie a logistika, ktorý vyučujem na Katedre kybernetiky a umelej inteligencie, Fakulty elektrotechniky a informatiky, Technickej univerzity v Košiciach od roku 1998. Tento predmet je zaradený v bakalárskych študijných programoch v rámci viacerých študijných odborov – Hospodárska informatika, Kybernetika a Inteligentné systémy.

Cieľom týchto podkladov je poskytnúť našim študentom stručný, ale ucelený pohľad na látku preberanú v predmete Rozvrhovanie a logistika.

Problematikou rozvrhovania som sa zaoberal vo svojej dizertačnej práci, kde som navrhol a experimentálne overil niektoré nové prístupy pri riešení úloh rozvrhovania (typu „job shop“) na báze logického programovania ohraničení. Preto rozvrhovanie tvorí jednu z nosných častí tohoto predmetu. Úlohy rozvrhovania sú však v predmete zasadené do širšieho kontextu výrobnej logistiky, v rámci ktorého sa v tomto predmete riešia okrem úloh rozvrhovania aj viaceré ďalšie typické optimalizačné úlohy, ako napr. distribučný problém.

Dôraz je pritom kladený na základné metódy, ktoré sa veľmi často využívajú pre riešenie optimalizačných úloh v rozvrhovaní a logistike, akými sú prostriedky lineárneho a celočíselného programovania, úlohy s ohraničeniami (ako všeobecnejší koncept), metódy umelej inteligencie, alebo rôzne heuristiky.

Dúfam, že túto publikáciu ocenia najmä naši študenti v spomínaných bakalárskych študijných programoch Hospodárska informatika, Kybernetika a Inteligentné systémy na FEI, TU v Košiciach.

Moje poďakovanie na tomto mieste patrí recenzentovi za starostlivé prečítanie rukopisu, opravu viacerých formálnych, ale aj vecných chýb a za cenné pripomienky a námety, ktoré obohatili obsah predkladaného textu.

Košice, máj 2010 autor

1 Úvod

V prvej kapitole budú vysvetlené základné pojmy, najmä vybrané definície logistiky, jej ciele a predstavená bude aj štruktúra základných činností v procese výrobnej logistiky.

1.1 Logistika

Keď sa pozrieme na pojem logistika z etymologického hľadiska, význam tohto slova vychádza z gréckych výrazov:

- logo – myslieť, slovo
- logos – súdny, mysliaci
- logistika – praktické umenie počítať

Ako odborný termín však tento pojem použil prvý krát v roku 1837 švajčiarsky generál Jomini vo svojej práci „Náčrt vojenského umenia“, kde chápe logistiku ako vedu o pohybe, zásobovaní a ubytovaní vojenských jednotiek. Od 20. storočia sa tento pojem používa aj v civilnej sfére, najmä v hospodárstve v súvislosti s plánovaním a riadením hmotných aj informačných tokov pri zabezpečení výroby a distribúce tovarov a služieb.

Hlavným cieľom logistiky je minimalizácia celkových nákladov v spomínanom výrobnom a distribučnom procese. Z množstva definícií logistiky si dovolím vybrať dve. Jednu podľa [Malindžák 1996], ktorý logistiku definuje ako *ucelenú teóriu o spôsoboch zabezpečenia plynulého toku tovaru a informácií s cieľom minimalizácia nákladov*.

Druhá definícia pochádza z amerického prostredia [Lambert et al. 2000] a logistiku chápe ako *tú časť procesu zásobovacieho reťazca (supply chain), ktorá plánuje, implementuje a riadi efektívny tok a skladovanie tovarov, služieb a súvisiacich informácií, medzi bodom vzniku a bodom spotreby s cieľom uspokojiť požiadavky zákazníka.*

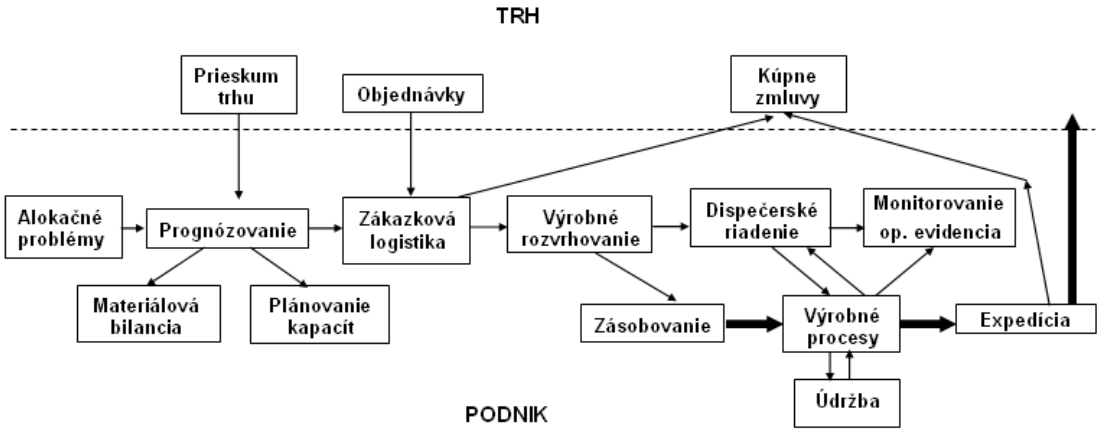
Existuje viacero špecifických oblastí logistiky, napr. [Dupaľ, Brezina 2005] rozdeľujú podnikovú logistiku na nákupnú a zásobovaciu logistika, výrobnú logistiku a distribučnú logistiku. V ďalšom však budeme vychádzať z výrobnej logistiky a jej štruktúry definovanej v nasledujúcej časti **Chyba! Nenašiel sa žiaden zdroj odkazov.**.

1.2 Štruktúra činností výrobnej logistiky

Štruktúru základných činností výrobnej logistiky uvádzam na Obr. 1, ktorá je inšpirovaná publikáciou [Malindžák 1996].

Ako prvá z činností je uvedené ***riešenie alokačných úloh***. Existuje mnoho úloh tohoto typu. Dôvod, prečo je tento druh úloh zaradený na začiatok celého reťazca výrobnej logistiky je ten, že jednou z alokačných úloh je rozhodnutie, ktoré je potrebné prijať hneď na začiatku – o umiestnení výroby, kde spravidla existuje viacero alternatív. Inou úlohou tohto druhu je napr. rozhodnutie o umiestnení distribučných centier a pod. Alokačným úlohám je venovaná celá kapitola 3 tohto učebného textu.

Ďalšou činnosťou je ***prognózovanie***, ktorého cieľom je najmä stanovenie odhadu predaja výrobkov v nasledujúcom obdobie, ktorý je kľúčový pre plánovanie kapacít výroby na nasledujúce obdobie. Prognózovaniu je venovaná kapitola 4. Jedným typom kapacít sú vstupné suroviny a materiál. Blok ***materiálová bilancia*** má za cieľ stanovenie potrebného množstva materálov na výrobu a jeho následné objednanie.



Obr. 1 Štruktúra činností výrobnej logistiky

Ďalšou dôležitou činnosťou a zároveň špeciálnym typom logistiky je ***zákazková logistika***, ktorej cieľom je riadenie toku objednávok vo firme od ich evidencie až po podpis (potvrdenie) kúpnych zmlúv a ich prípravu pre operatívne plánovanie. To zahŕňa aj technické a ekonomické posúdenie objednávok. Podrobnejšie sa tejto problematike venuje napr. už spomínaná publikácia [Malindžák 1996].

Výrobné rozvrhovanie definuje čo, kedy a kde sa bude vyrábať, t.j. má na starosti časové plánovanie výroby. Rozvrhovaniu sa podrobne venuje kapitola 6 tohto učebného textu. Plán výroby predstavuje vlastne definíciu cieľov pre následné *dispečerské riadenie*, ktoré dozerá na realizáciu výrobných plánov vo *výrobnom procese*. *Monitorovanie procesov a operatívna evidencia* predstavujú podporný nástroj a pomoc pre dispečera pri riadení výroby. Posledne spomínané činnosti spadajú skôr do oblasti riadenia výroby a nie sú priamou súčasťou typických logistických činností.

Naopak ***zásobovanie*** je jednou z typických logistických činností. Cieľom zásobovania je riadenie stavu skladových zásob pre zabezpečenie plynulej výroby a minimalizácia strát plynúcich z nadbytočných zásob. Vybraným modelom riadenia skladových zásob je venovaná posledná kapitola 7 tohto učebného textu.

Logistické činnosti ešte dopĺňa ***údržba***, t.j. plánovanie pravidelnej údržby a realizácia opráv výrobných zariadení. Teóriu procesov obnovy, ktoré patria do tejto časti logistiky môže čitateľ nájsť napr. v publikácii [Dudorkin 1997]. Nakoniec ***expedícia*** má na starosti balenie a dodávku výsledného produktu zákazníkovi.

1.3 Metódy používané na riešenie úloh v logistike

Na riešenie úloh, ktoré vznikajú v rámci jednotlivých blokov štruktúry činností výrobnej logistiky (viď. Obr. 1) sa používajú rôzne metódy. Tieto možno rozdeliť rôznym spôsobom. V ďalšom budeme metódy rozdeľovať na tri skupiny nasledovne.

Metódy operačného výskumu sú jedným z hlavných zdrojov rôznych matematických modelov, ktorými je možné veľmi dobre reprezentovať mnohé logistické úlohy a algoritmy na riešenie týchto úloh.

Metódy umelej inteligencie slúžia najmä na riešenie rôznych optimalizačných úloh, u ktorých nie sú známe polynomiálne algoritmy a teda ich riešenie si vyžaduje čas, ktorý rastie exponenciálne s veľkosťou vstupu danej úlohy. Pritom metódy, ktoré pochádzajú z oblasti umelej inteligencie, sú veľmi rôznorodé, napr. metódy na riešenie úloh s ohraničeniami, genetické algoritmy a ďalšie.

Na riešenie mnohých logistických úloh boli postupne vytvorené rozličné **heuristiky**, ktoré sú výsledkom ľudskej šikovnosti, schopnosti premýšľať a navrhovať účinné postupy na riešenie zložitých úloh.

2 Operačný výskum

Operačný výskum (alebo aj operačná analýza [Madarász et al. 2008]) je vedecká disciplína, predmetom ktorej je skúmanie operácií v organizačných jednotkách [Dudorkin 1997]. Pod operáciou pritom chápeme postupnosť vzájomne závislých akcií smerujúcich k určitému cieľu. Cieľom operačného výskumu je vypracovať také závery a doporučenia, ktoré slúžia ako podklad pre čo najlepšie riadenie skúmaných operácií. V práci [Madarász et al. 2008] je táto vedecká disciplína zahrnutá medzi súbor vied, ktoré sa zaoberajú problematikou systémov, ako jedna z troch hlavných oblastí podľa autormi zavedenej klasifikácie.

V rámci operačného výskumu vznikli mnohé matematické modely, ktoré sú účinnými nástrojmi na reprezentáciu a riešenie mnohých logistických úloh. Tieto matematické modely možno rozdeliť podľa rôznych hľadísk.

Jedným takýmto hľadiskom môže byť to, či sa v týchto modeloch vyskytujú náhodné veličiny. Z tohto hľadiska možno potom matematické modely rozdeliť na dve skupiny:

1. Stochastické modely – obsahujú náhodné veličiny
2. Deterministické modely – neobsahujú náhodné veličiny

Ďalším hľadiskom je to, či tieto modely zachytávajú aj zmeny v čase, t.j. či obsahujú časové veličiny. Z tohto pohľadu máme opäť dve skupiny modelov:

1. Dynamické - obsahujú (modelujú) časové zmeny
2. Statické - neobsahujú (nemodelujú) časové zmeny

Iným hľadiskom je to, či dané matematické modely obsahujú kriteriálnu funkciu, ktorej hodnotu sa snažia maximalizovať, resp. minimalizovať. Podľa tohto hľadiska sú tiež dve skupiny modelov:

1. Rozhodovacie - obsahujú kriteriálnu funkciu – optimalizujú
2. Technologické - neobsahujú kriteriálnu funkciu – neoptimalizujú

Z početných modelov operačného výskumu sa veľmi často používajú lineárne a celočíselné programovanie. Ide o rozhodovacie modely (optimalizačné), ktoré sú väčšinou statické a deterministické.

2.1 Lineárne programovanie

Je to riešenie optimalizačnej (extremálnej) úlohy, tzv. problému lineárneho programovania. Tento problém spočíva v hľadaní takej n-tice reálnych čísel $\bar{x}^T = (x_1, x_2, ..., x_n)$,

pre ktorú nadobúda kriteriálna funkcia $f(\bar{x}) = c_1 x_1 + c_2 x_2 + ... + c_n x_n$

minimum alebo maximum, a ktorá spĺňa obmedzujúce podmienky v tvare:

$a_{i1} x_1 + a_{i2} x_2 + ... + a_{in} x_n \geq b_i$ (i = 1, ... , m),

prípadne aj podmienky nezápornosti

$x_j \geq 0$ (j = 1, ... , n).

Koeficienty a_{ij}, b_i a c_i v predchádzajúcich výrazoch sú všetko konštanty

(presne dané reálne čísla). Pre ilustráciu uvažujme nasledujúci príklad. Mimochodom, ide o tzv. ***úlohu o výrobnom programe***, ktorá patrí k jedným z typických logistických úloh.

Príklad

Závod vyrába 2 rôzne kusové výrobky v 2 technologických procesoch. Na výrobu prvého výrobku sú potrebné 3kg suroviny S_1 a 1kg suroviny S_2. Na výrobu druhého výrobku sú potrebné 3kg suroviny S_1 a 2kg suroviny S_2. Závod má zásoby prvej suroviny S_1 = 2100kg a druhej suroviny S_2 = 1000kg. Cieľom úlohy je zostrojiť taký plán, v ktorom maximalizujeme zisk z produkcie. Sumár zadaných údajov je uvedený na Obr. 2.

-	$S_1\ \left[\frac{kg}{ks}\right]$	$S_2\ \left[\frac{kg}{ks}\right]$	$cena\ \left[\frac{PJ}{ks}\right]$
1. výrobok	3	1	2
2. výrobok	3	2	3
zásoby [kg]	2100	1000	-

Obr. 2 Zadanie príkladu úlohy o výrobnom programe (2 výrobky, 2 suroviny)

Riešenie:

Nech x_i (i = 1, 2) je počet výrobkov i-tého typu, t.j. x_1 je počet výrobkov prvého typu, ktoré je potrebné vyrobiť a x_2 je počet výrobkov druhého typu, ktoré je potrebné vyrobiť. Nakoľko obe premenné vyjadrujú počet výrobkov, ich hodnoty prirodzene musia byť nezáporné, t.j. platí že $x_1, x_2 \geq 0$.

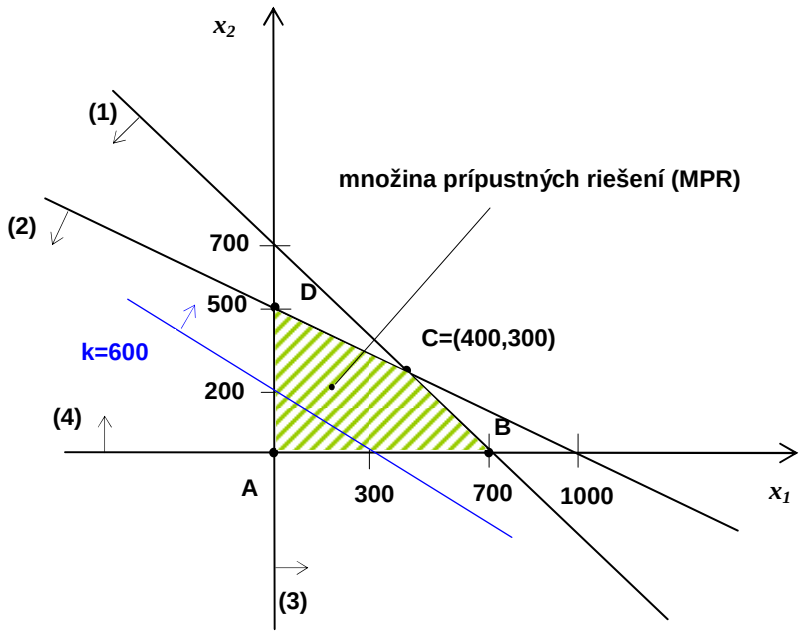
Cieľom je dosiahnuť maximálny zisk z výroby. Celkový zisk z výroby bude daný hodnotou funkcie $f(\bar{x}) = 2x_1 + 3x_2$, ktorú chceme maximalizovať.

Ohraničujúce podmienky vyplývajú z limitovaných zásob surovín na sklade. Pre každú zo surovín vstupujúcich do výrobného procesu budeme mať jednu ohraničujúcu podmienku, ktorá musí byť splnená. Konkrétne pre surovinu S_1 bude táto podmienka vyjadrená nerovnicou $3x_1 + 3x_2 \leq 2100$. Analogicky pre druhú surovinu S_2 bude táto podmienka daná nerovnicou $x_1 + 2x_2 \leq 1000$.

Pre prípad dvoch výrobkov možno túto úlohu ľahko vyriešiť graficky (viď. Obr. 3). Grafický postup pri riešení úloh lineárneho programovania je nasledovný. Každá z dvoch súradnicových osí, ktoré zakreslíme, bude zodpovedať jednej z dvoch premenných (v našom príklade teda x_1 a x_2). Potom do grafu zakreslíme každú zo zadaných ohraničujúcich podmienok. V našom prípade sú také dve (pre každú surovinu jedna nerovnica), ďalšie dve vyplývajú z nezápornosti oboch premenných. Spravidla každá nerovnica popisuje v takto zostrojenom dvojrozmernom priestore nejakú polrovinu, ktorej hraničnú priamku a príslušnú časť roviny potrebujeme určiť.

Nezápornosť premenných x_1 a x_2 znamená, že riešenie sa musí nachádzať v prvom kvadrante, vymedzenom na Obr. 3 polrovinami *(3)* pre $x_1 \geq 0$ (t.j. hraničnou priamkou je os x_2 a príslušná časť roviny leží napravo od tejto hraničnej priamky) a *(4)* pre $x_2 \geq 0$ (t.j. hraničnou priamkou je os x_1 a príslušná časť roviny leží nahor od tejto hraničnej priamky).

Ohraničujúcu podmienku vyplývajúcu z obmedzných zásob suroviny S_1 máme vyjadrenú nerovnicou $3x_1 + 3x_2 \leq 2100$. Hraničnú priamku polroviny, ktorá zodpovedá bodom vyhovujúcim tejto podmienke zostrojíme tak, že nájdeme dva ľubovoľné body vyhovujúce rovnici $3x_1 + 3x_2 = 2100$. Stačí ak si zvolíme ľubovoľnú hodnotu premennej x_1 a dopočítame x_2 , resp. naopak. Výpočet bude najjednoduchší, ak si zvolíme hodnotu $x_1 = 0$, potom $3 \cdot 0 + 3x_2 = 2100$ => $3x_2 = 2100$ => $x_2 = 700$. Takže hraničná priamka *(1)* určite prechádza bodom [0,700]. Druhý bod môžme získať podobne, ak pre zmenu zvolíme hodnotu $x_2 = 0$, potom $3x_1 + 3 \cdot 0 = 2100$ => $3x_1 = 2100$ => $x_1 = 700$. Takže hraničná priamka *(1)* určite prechádza aj bodom [700,0]. Teraz už vieme túto priamku jednoducho zakresliť a ostáva už iba zistiť, ktorá polrovina vyhovuje zadanej ohraničujúcej podmienke $3x_1 + 3x_2 \leq 2100$. To zistíme tak, že do nerovnice dosadíme hodnoty premenných zodpovedajúce ľubovoľnému bodu v rovine mimo deliacej priamky *(1)*. Ak súradnice tohto bodu vyhovujú nerovnici, potom treba vyznačiť práve polrovinu, v ktorej leží zvolený bod. Ak nevyhovuje, potom treba vyznačiť tu druhú polrovinu, v ktorej daný bod neleží. Pre jednoduchosť výpočtu zvoľme bod [0,0].



Obr. 3 Grafické riešenie úlohy o výrobnom programe pre 2 výrobky

Po dosadení do nerovnice $3 \cdot 0 + 3 \cdot 0 \leq 2100$ => $0 \leq 2100$ zistíme, že vyhovuje nerovnici, takže v našom dvojozmenrom priestore vyznačíme polrovinu ležiacu naľavo a nadol od deliacej priamky *(1)*.

Analogicky pre druhú surovinu S_2 máme ohraničujúcu podmienku danú nerovnicou $x_1 + 2x_2 \leq 1000$. Hraničnú priamku polroviny, ktorá zodpovedá bodom vyhovujúcim tejto podmienke zostrojíme tak, že nájdeme dva ľubovoľné body vyhovujúce rovnici $x_1 + 2x_2 = 1000$. Opäť zvolíme ľubovoľnú hodnotu premennej x_1 a dopočítame x_2, resp. naopak, aby sme našli dva

ľubovoľné body ležiace na tejto priamke a mohli ju zostrojiť. Ak si zvolíme hodnotu $x_1 = 0$, potom $0 + 2x_2 = 1000$ => $2x_2 = 1000$ => $x_2 = 500$. Takže hraničná priamka *(2)* určite prechádza bodom [0,500]. Druhý bod môžme získať podobne, ak zvolíme napr. hodnotu $x_2 = 0$, potom $x_1 + 2 \cdot 0 = 1000$ => $x_1 = 1000$ => $x_1 = 1000$. Takže hraničná priamka *(2)* určite prechádza aj bodom [1000,0]. Priamku zakreslíme a ostáva už iba zistiť, ktorá polrovina vyhovuje zadanej ohraničujúcej podmienke $x_1 + 2x_2 \leq 1000$. Do nerovnice dosadíme hodnoty premenných zodpovedajúce ľubovoľnému bodu v rovine mimo deliacej priamky *(2)*, napr. opäť bod [0,0]. Po dosadení do nerovnice $0 + 2 \cdot 0 \leq 1000$ => $0 \leq 1000$ zistíme, že vyhovuje nerovnici, takže v našom dvojozmenrom priestore vyznačíme polrovinu ležiacu naľavo a nadol od deliacej priamky *(2)*.

Takzvaná **množina prpustných riešení (MPR)** je daná prienikom všetkých polrovín zodpovedjúcich jednotlivým ohraničujúcim podmienkam, t.j. v našom príklade máme štyri takéto polroviny ohraničené deliacimi priamkami *(1), (2), (3)* a *(4)*. Výsledkom je štvoruholník ABCD. Vlastnosťou MPR je, že všetky body z MPR vyhovujú všetkým ohraničujúcim podmienkam danej úlohy lienárneho programovania a zodpovedajú teda platným riešeniam. Zostáva otázka, v ktorom z týchto bodov nadobúda daná kriteriálna funkcia svoj extrém (minimum, resp. maximum).

V našom príklade teda každý bod ležiaci v štvoruholníku ABCD zodpovedá prípustnému výrobnému programu, t.j. hodnoty súradníc x_1 a x_2 sú také hodnoty, že na výrobu zodpovedajúcich množstiev výrobkov prvého a druhého typu bude dostatočné množstvo surovín. Ale každý z týchto bodov môže predstavovať inú hodnotu zisku. Na to, aby sme našli optimálne riešenie úlohy o výrobnom programe, musíme nájsť práve taký bod ležiaci v štoruholníku ABCD, pre ktorý hodnota kriteriálnej funkcie $f(\bar{x}) = 2x_1 + 3x_2$ nadobúda maximum.

Dá sa dokázať, že to musí byť niektorý z krajných bodov tohto štvoruholníka. Takže by sme mohli vypočítať hodnoty kriteriálnej funkcie zodpovedajúcej každému z týchto bodov a zistiť, kde je jej hodnota najvyššia. Iná možnosť je zostrojiť si parametrickú priamku pre určitú hodnotu zisku (t.j. priamku, ktorej všetky body ležiace v MPR predstavujú riešenia s rovnakou hodnotou kriteriálnej funkcie). Zostrojme napr. parametrickú priamku pre hodnotu parametra k = 600, t.j. hodnota kriteriálnej funkcie $f(\bar{x}) = 2x_1 + 3x_2 = 600$. Priamku zodpovedajúcu tejto rovnici zostrojíme rovnakým spôsobom, ako bolo uvedené vyššie, t.j. nájdeme jej ľubovoľné dva body a nimi vedieme priamku. Zvolíme ľubovoľnú hodnotu premennej x_1 a dopočítame x_2, resp. naopak. Ak si zvolíme hodnotu $x_1 = 0$, potom $2 \cdot 0 + 3x_2 = 600$ => $3x_2 = 600$ => $x_2 = 200$. Takže parametrická priamka pre hodnotu zisku k = 600 určite prechádza bodom [0,200]. Druhý bod môžme získať podobne, ak zvolíme napr. hodnotu $x_2 = 0$, potom $2x_1 + 3 \cdot 0 = 600$ => $2x_1 = 600$ => $x_1 = 300$. Takže parametrická priamka pre k = 600 určite prechádza aj bodom [300,0]. Ľahko zistíme, že zvyšovanie parametra k (čo zodpovedá vyšším hodnotám zisku z produkcie definovanej zodpovedajúcimi riešeniami úlohy o výrobnom programe) zodpovedá rovnobežkám parametrickej priamky smerom nahor

od zostrojenej parametrickej priamky pre k = 600. Maximum zisku teda bude zodpovedať rovnobežke s touto priamkou, ktorá bude prechádzať bodom C.

Teraz nám už ostáva iba určiť súradnice bodu C, zodpovedajúceho optimálnemu riešeniu zadanej úlohy o výrobnom programe. Ide o priesečník priamok *(1)* a *(2)*, takže súradnice získame riešením sústavy týchto dvoch rovníc o dvoch neznámych.

$$3x_1 + 3x_2 = 2100$$
$$x_1 + 2x_2 = 1000$$

Ak z druhej rovnice vyjadríme x_1 a dosadíme do prvej, dostaneme:

$$x_1 = 1000 - 2x_2$$
$$3 \cdot (1000 - 2x_2) + 3x_2 = 2100$$
$$-3 \cdot x_2 = 2100 - 3000$$
$$x_2 = 300$$

Spätným dosadením získanej hodnoty x_2 do druhej rovnice dopočítame zodpovedajúcu hodnotu x_1, t.j.

$$x_1 + 2 \cdot 300 = 1000$$
$$x_1 = 400$$

Optimálne riešenie danej úlohy o výrobnom programe je teda x_1 = 400 a x_2 = 300, čomu zodpovedá výsledný zisk $f(\bar{x}) = 2 \cdot 400 + 3 \cdot 300 = 1700$.

2.1.1 Predpoklady modelov úloh lineárneho programovania

Úlohy lineárneho programovania predpokladajú niektoré všeobecne platné vlastnosti, ktoré sú tu stručne vysvetlené.

1. ***Proporcionalita***, t.j. predpoklad priamej úmernosti spotreby jednotlivých zdrojov a celej produkcie s veľkosťou produkcie.
2. ***Aditívnosť***, alebo predpoklad súčtu – celková spotreba ľubovoľného zdroja a celková cena sa rovná súčtu jednotlivých dielčích spotrieb a dielčích cien vyplývajúcich z produkcie v jednotlivých procesoch.
3. ***Predpoklad deliteľnosti***, t.j. že pripúšťame k riaditeľným premenným nielen celočíselné, ale aj zlomkové hodnoty.
4. ***Predpoklad nezápornosti***, t.j. že spravidla nie sú prípustné záporné hodnoty riadiacich premenných.

2.1.2 Normovaný tvar úloh lineárneho prgramovania

Niektoré metódy a softvérové nástroje na riešenie úloh lineárneho programovania predpokladajú na vstupe zadanú ***úlohu lineárneho programovania v tzv. normovanom tvare***. Normovaný tvar musí spĺňať nasledovné podmienky:

1. Cieľom je minimalizácia danej kriteriálnej funkcie
2. Všetky ohraničujúce podmienky sú zadané v tvare rovností

3. Všetky premenné sú nezáporné

$$f(\bar{x}) \overset{!}{=} MIN$$
$$\forall i = 1..m : a_{i1}x_1 + a_{i2}x_2 + ... + a_{in}x_n = b_i$$
$$\forall j = 1..n : x_j \geq 0$$

Previesť ľubovoľne ináč zadanú úlohu lineárneho programovania na normovaný tvar však nie je problémom. Pritom môžu nastať rozdiely oproti normovanému tvaru, ktoré je potrebné vhodnými transformáciami riešiť.

1. Ak je cieľom maximalizácia kriteriálnej funkcie $f(\bar{x})$, potom stačí pri prevode na normovaný tvar vynásobiť túto kriteriálnu funkciu (-1), čím sa vlastne zmení úloha maximalizácie funkcie $f(\bar{x})$ na minimalizáciu funkcie $g(\bar{x}) = -f(\bar{x})$. T.j. $Ak: f(\bar{x}) \overset{!}{=} MAX \Rightarrow g(\bar{x}) = -f(\bar{x}) \Rightarrow g(\bar{x}) \overset{!}{=} MIN$.
2. Ak kriteriálna funkcia obsahuje aj nejakú konštantu, t.j. $f(\bar{x}) = g(\bar{x}) + k$, potom stačí uvažovať kriteriálnu funkciu bez započítania konštanty, ktorá nemení usporiadanie riešení vzhľadom na kriteriálnu funkciu, predstavuje len posun ich hodnoty stále o rovnakú konšantu k. T.j. $MIN(g(\bar{x}) + k) = MIN(g(\bar{x})) + k$.
3. Ak ohraničujúca podmienka je v tvare nerovnosti, potom je potrebné zaviesť pre každú takúto ohraničujúcu podmienku jednu novú, pomocnú premennú (anglicky sa nazývajú takéto premenné „slack variables“) a túto buď pripočítať, alebo odpočítať, podľa orientácie znaku nerovnosti. T.j.

$$Ak.: a_{i1}x_1 + a_{i2}x_2 + ... + a_{in}x_n \geq b_i \Rightarrow pomocná \quad premenná \quad x_{n+1} \geq 0:$$
$$a_{i1}x_1 + a_{i2}x_2 + ... + a_{in}x_n - x_{n+1} = b_i$$

$$Ak: a_{i1}x_1 + a_{i2}x_2 + ... + a_{in}x_n \leq b_i \Rightarrow pomocná \quad premenná \quad x_{n+1} \geq 0$$
$$a_{i1}x_1 + a_{i2}x_2 + ... + a_{in}x_n + x_{n+1} = b_i$$

4. Ak niektorá z riaditeľných premenných môže nadobúdať aj záporné hodnoty, potom pre každú takúto premennú je potrebné zaviesť dve nové pomocné premenné (obe nezáporné) takým spôsobom, že platí:

$$Ak: x_j \leq 0 \Rightarrow pomocné \quad premenné \quad x_k, x_l \geq 0:$$
$$x_j = x_k - x_l$$

2.1.3 Ďalšie vlastnosti úloh lineárneho prgramovania

V tejto časti uvedieme niektoré ďalšie vlastnosti úloh lineárneho programovania, ktoré sú kľúčové pre pochopenie princípu simplexovej metódy, ktorá sa najčastejšie používa na riešenie úloh lineárneho programovania.

1. Množinu ohraničujúcich podmienok normovanej úlohy lineárneho

programovania je možné vyjadriť aj ***v maticovom tvare***: $\overline{\overline{A}x} = \overline{b}$.

2. Maticové vyjadrenie sa dá zase ľahko prepísať do ***vektorového tvaru***, ktorý vyzerá nasledovne: $x_1\overline{P_1} + x_2\overline{P_2} + ... + x_n\overline{P_n} = \overline{P_0}$.

   Pritom $\overline{P_j} = \begin{pmatrix} a_{1j} \\ ... \\ a_{mj} \end{pmatrix}$, $\forall j = 1..n$

3. Medzi riešeniami sústavy rovníc, zodpovedajúcej maticovému, resp. vektorovému vyjadreniu normovanej úlohy lineárneho programovania majú kľúčové postavenie tzv. ***bázické riešenia***, t.j. také, v ktorých práve *m* premenných (t.j. práve toľko, koľko je zadaných ohraničujúcich podmienok v tvare rovníc) má nenulové hodnoty. Všetky ostatné premenné nadobúdajú nulové hodnoty, t.j. napr. $\overline{x} = (x_1, x_2, ..., x_m, 0, ..., 0)$.
4. ***Množina prípustných riešení (MPR) je konvexná***. To znamená, že pre všetky dvojice prípustných riešení $\overline{x_1}, \overline{x_2}$ z MPR platí, že aj všetky riešenia ležiace na ich spojnici sú z MPR. Táto množina prípustných riešení preto vytvára v *n*-rozmernom priestore všetkých možných kombinácií hodnôt premenných konvexný polyéder. V prípade dvojrozmernej úlohy (*n* =2), ako tomu bolo v predchádzajúcom príklade ide o konvexný mnohouholník (napr. štvoruholník v našom prípade).
5. Riešenie úloh lineárneho programovania je zjednodušené tým, že ***kriteriálna funkcia nadobúda svoj extrém aspoň v 1 krajnom bode MPR***. To znamená, že nám vlastne stačí preskúmať konečný počet krajných bodov, t.j. vrcholov konvexného polyédra, ktorý predstavuje MPR.
6. Hľadanie krajných bodov MPR je ekvivalentné hľadaniu prípustných bází sústavy vektorov $(\overline{P_1}, \overline{P_2}, ..., \overline{P_n})$. T.j. každé bázické riešenie množiny rovníc vyjadrujúcich ohraničujúce podmienky normovanej úlohy lineárneho programovania zodpovedá vrcholu konvexného polyédra reprezentujúceho MPR danej úlohy.

2.1.4 Princíp simplexovej metódy

Na základe vlastností úloh lineárneho programovania popísaných v predchádzajúcej časti môžme teraz pristúpiť k sformulovaniu princípu činnosti, resp. hrubému náčrtu simplexového algoritmu, ktorý sa úspešne používa na riešenie úloh lineárneho programovania. Tento algoritmus je založený na efektívnom prieskume krajných bodov MPR.

---

1. Vyjdeme z ľubovoľného krajného bodu (vrcholu MPR), ktoré zodpovedá nejakému bázickému riešeniu sústavy lineárnych rovníc normovaného tvaru úlohy lineárneho programovania.

2. Prejdeme k takému krajnému bodu MPR, ktorého hodnota kriteriálnej funkcie je lepšia. Prechod od jedného

```
        krajného bodu k inému, znamená prechod od jedného
        bázického riešenia k inému bázickému riešeniu.

     3. Tento krok sa opakuje, až kým už neexistuje krajný
        bod s lepšou hodnotou kriteriálnej funkcie. Potom
        aktuálne nájdené bázické riešenie je optimálne.
```

2.2 Celočíselné programovanie

Pri úlohách definovaných v predchádzajúcej kapitole môže byť požiadavka na definičný obor premenných iná než je tomu u lineárneho programovania, ktoré požaduje ako definičný obor reálne čísla. Zoberme si len príklad z predchádzajúcej kapitoly, úlohu o výrobnom rograme. Ak sa vyrábajú kusové výrobky, potom má zmysel uvažovať len o celočíselných hodnotách premenných vyjadrujúcich ich počet (resp. prirodzených číslach).

Pokiaľ ale zmeníme definičný obor z reálnych čísel napr. na celé čísla, prestáva platiť predpoklad deliteľnosti (viď. 2.1.1), čo má vážne dôsledky na riešenie takejto úlohy. MPR totiž prestáva byť súvislá (je tvorená izolovanými bodmi) a jednoduché riešenie úlohy simplexovým algoritmom a následné zaokrúhlenie výsledných hodnôt riaditeľných premenných nezaručuje nájdenie optimálneho riešenia (takto získané riešenie dokonca nemusí byť ani prípustné).

Vo všeobecnosti sú tri možnosti s ohľadom na doménu (definičný obor) premenných v danej optimalizačnej úlohe. Nech n je počet všetkých premenných v danej úlohe a n' je počet celočíselných premenných v tejto úlohe.

1. Ak $n'<n$, potom ide o tzv. ***zmiešané úlohy*** („mixed integer programming“)
2. Ak $n' = n$ **,** potom ide o čisté ***celočíselné programovanie***,
3. Ak $n' = n$, pričom $x_i \in \{0, 1\}$ $(i = 1, \ldots , n)$**,** potom ide o tzv. **bivalentné programovanie.**

Úlohy bivalentného programovania sú špeciálnym prípadom úloh celočíselného programovania a je možné ich riešiť rovnakými metódami. Metód je viacero, ale všetky možno rozdeliť do dvoch základných skupín:

1. ***Úplné metódy*** (napr. metóda vetvenia a medzí alebo metóda sečných nadrovín) – zaručujú nájdenie optimálneho riešenia, ale v nepolynomiálnom čase v závislosti od veľkosti úlohy (tj. počtu premenných n).
2. **Približné metódy** (napr. simulované žíhanie, genetické algoritmy) – tieto metódy nikdy nezaručia nájdenie optimálneho riešenia, ale v rozumnom čase poskytnú celkom dobré („suboptimálne“) riešenie.

V nasledujúcej časti bude vysvetená metóda vetvenia a medzí, ktorú je možné použiť na riešenie celého radu logistických a rozvrhovacích úloh.

2.2.1 Metóda vetvenia a medzí

Metóda vetvenia a medzí je založená na dvoch základných princípoch, ktoré sú premietnuté aj do jej názvu.

1. ***Princíp vetvenia*** – množinu prípustných riešení (MPR) rozkladá na radu spravidla disjunktných podmnožín. Tento princíp je známy napr. aj z umelej inteligencie ako „rozdeľuj a panuj“, pričom dochádza k dekompozícii úlohy na menšie, ľahšie riešiteľné časti, spravidla úlohy toho istého typu, takže možno využiť rekurzívne volanie funkcie alebo procedúry pri použití napr. procedurálneho programovania, resp. predikátu pri použití logického programovania na implementáciu metódy založenej na tomto princípe.
2. ***Princíp odhadu medzí*** – ide o odhady hodnoty kriteriálnej funkcie na MPR, resp. na niektorej jej podmnožine (tzv. horná medza a dolná medza). Odhadované medze sa používajú na tzv. orezávanie, t.j. celú podmnožinu nie je nutné skúmať ďalej, ak odhad najlepšieho možného riešenia v nej nedosahuje kvalitu aktuálne najlepšieho nájdeného riešenia v procese riešenia úlohy.

Postup metódy vetvenia a medzí pre prípad maximalizácie kriteriálnej funkcie vyzerá nasledovne.

---

1. Inicializácia - uvažujeme celú MPR ako jediného kandidáta vetvenia. Stanovíme dolnú (spodnú) medzu kriteriálnej funkcie $f_S = -\infty$ (*ide o najlepšie doteraz nájdené riešenie, platí globálne*) a hornú medzu kriteriálnej funkcie f_H (*platí len pre danú množinu riešení, v tomto prípade celé MPR*) odhadneme v závislosti od charakteru úlohy (*najväčšia teoreticky možná hodnota kriteriálnej funkcie pre riešenie z MPR*).

2. Vetvenie – podľa vybraného pravidla vetvenia (*napr. najvyššia hodnota f_H*), vyberieme z kandidátov vetvenia jednu množinu a rozložíme ju na jednu alebo viac podmnožín prípustných riešení.
   - Ak je súbor kandidátov prázdny, tak algoritmus končí. Pritom riešenie zodpovedajúce aktuálnej spodnej medzi f_S je optimálne. T.j. $f_S = f(\bar{x})$ => $\bar{x}$ je optimálne riešenie.
   - Ak $f_S = -\infty$, potom riešenie neexistuje (MPR je prázdna).

3. Stanovenie hornej medze f_H – pre každú novú podmnožinu stanovíme hornú medzu kriteriálnej funkcie na tejto podmnožine (*najlepšia teoreticky možná hodnota kriteriálnej funkcie pre riešenia z danej podmnožiny*).

4. Orezávanie – z ďalšieho skúmania vylúčime tie podmnožiny, pre ktoré $f_H < f_S$, alebo ktoré sú prázdne.

5. Stanovenie spodnej medze f_S – určíme najlepšie prípustné riešenie $\bar{x}$ pre každú novú podmnožinu:
   - Ak $f(\bar{x}) \geq f_S$ => upravíme aktuálnu dolnú medzu $f_S = f(\bar{x})$,

$\bar{x}$ je najlepšie doteraz nájdené riešenie a opätovne sa vykoná orezávanie, pričom sa zistí či nemožno vylúčiť ďalšie podmnožiny na základe podmienky z kroku 4).

6. Pokračuj krokom 2.

Príklad

V skúmanom období môže podnik prevziať 6 rôznych zákaziek, ktoré sa líšia spotrebou času a materiálu. Kapacity výrobného zariadenia a zásob materiálu sú obmedzené. Úlohou je rozhodnúť, ktoré zákazky má podnik prevziať. Sumár zadaných údajov je uvedený na Obr. 4.

Zakazka	1	2	3	4	5	6	Kapacita
Zisk	11	63	9	5	4	8	-
Cas	1	7	1	1	1	5	15
Material	15	70	10	5	3	2	100

Obr. 4 Zadanie príkladu úlohy o výbere zákaziek (6 zákaziek, obmedzené časové a materiálové možnosti)

Riešenie:

Njaprv musíme zostaviť matematický model tejto úlohy. Každej potenciálnej zákazke priradíme jednu premennú x_i, i = 1..6. Definičný obor každej z týchto premenných bude dvojhodnotový, t.j. premenná nadobudne hodnotu x_i = 1, ak zákazka bude prijatá, v opačnom prípade bude hodnota x_i = 0. Takže $x_i \in \{0,1\}, \forall i = 1..6$. Kriteriálnu funkciu definujeme ako výsledný zisk z prijatých zákaziek, ktorý chceme maximalizovať, t.j.

$$f(\bar{x}) = 11x_1 + 63x_2 + 9x_3 + 5x_4 + 4x_5 + 8x_6 \stackrel{!}{=} MAX$$

Ohraničujúce podmienky budú dve, jedna vyjadrujúca obmedzený čas, ktorý má firma na spracovanie zákaziek, t.j. $x_1 + 7x_2 + x_3 + x_4 + x_5 + 5x_6 \leq 15$ a druhá zase obmedzené množstvo materálu, ktoré má firma k dispozícii, t.j. $15x_1 + 70x_2 + 10x_3 + 5x_4 + 3x_5 + 2x_6 \leq 100$.

Vzhľadom na charakter premenných ide o úlohu bivalentného programovania, ktoré je len špeciálnym prípadom celočíselného programovania. Preto na riešenie tejto úlohy použijeme metódu vetvenia a medzí. Postup budeme zakresľovať aj graficky vo forme stromu prehľadávania (viď. Obr. 5), pričom jeho koreňový uzol je MPR a bude výsledkom prvého, inicializačného kroku metódy vetvenia a medzí. Označíme ho poradovým číslom 0. Každé vetvenie bude predstavovať rozvetvenie aktuálneho uzla stromu prehľadávania na zodpovedajúce podpriestory (podmnožiny MPR) s príslušnými charakteristikami hornej a spodnej medze (f_H, f_S). Každá takto vytvorená podmnožina bude mať zodpovedajúci matematický model a svoje identifikačné číslo, ktoré bude zároveň poradovým číslom jej vytvorenia.

Takže úvodným krokom inicializácie vytvoríme už spomínaný koreňový uzol

s číslom 0, ktorému zodpovedá pôvodný celočíselný matematický model, t.j.

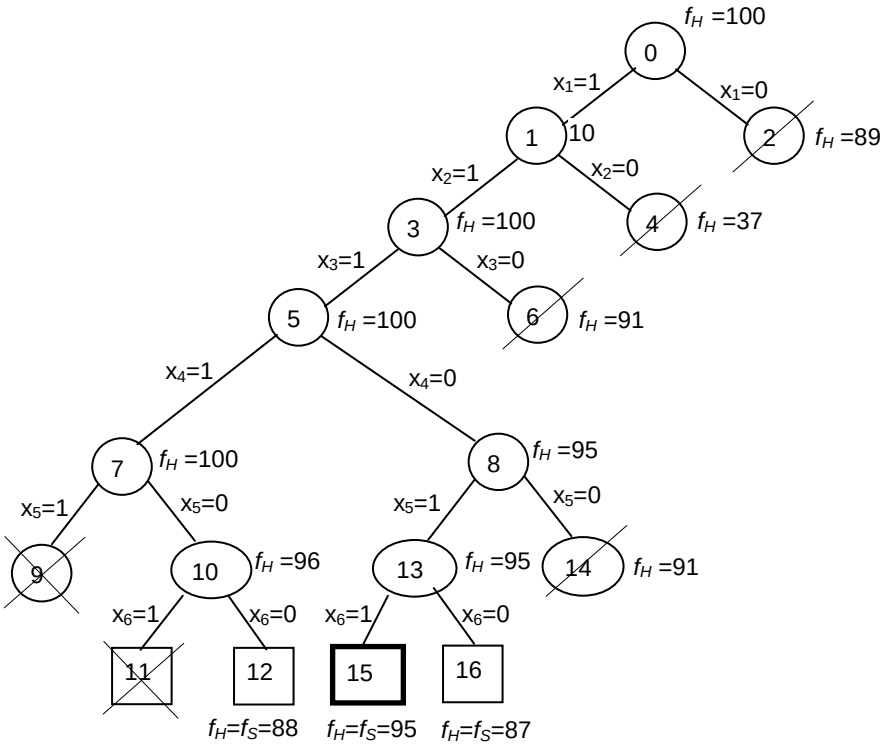
Uzol 0 *je charakterizovaný modelom:*

$$f(\bar{x}) = 11x_1 + 63x_2 + 9x_3 + 5x_4 + 4x_5 + 8x_6 \overset{!}{=} MAX$$
$$x_1 + 7x_2 + x_3 + x_4 + x_5 + 5x_6 \leq 15$$
$$15x_1 + 70x_2 + 10x_3 + 5x_4 + 3x_5 + 2x_6 \leq 100$$

Pre uzol 0 je potrebné teraz stanoviť hornú medzu, t.j. maximálnu teoreticky možnú hodnotu kriteriálnej funkcie. Tú získame tak, že ignorujeme ohraničujúce podmienky a všetkým premenným v kriteriálnej funkcii priradíme hodnotu 1 (čo zodpovedá prijatiu všetkých zákaziek). Takto získame najväčšiu teoreticky možnú hodnotu kriteriálnej funkcie $f_H(0) = 11 \cdot 1 + 63 \cdot 1 + 9 \cdot 1 + 5 \cdot 1 + 4 \cdot 1 + 8 \cdot 1 = 100$. To znamená, že hodnota kriteriálnej funkcie žiadneho riešenia nemôže byť vyššia ako 100.

Hodnotu spodnej medze nastavíme na $f_S(0) = -\infty$, t.j. každé riešenie, ktoré v ďalšom postupe nájdeme, bude mať určite lepšiu hodnotu kriteriálnej funkcie ako je $-\infty$.

V druhom kroku nasleduje vetvenie. Vetvenie vykonáme pridelením hodnoty jednej z premenných. V našom prípade budeme postupovať vo výbere premenných podľa ich poradia (neaplikujeme teda žiadnu heuristiku pre výber premenných, ktorá by mohla viesť k menšiemu stromu prehľadávania). Ako prvú teda vyberieme premennú x_1. Keďže môže nadobúdať dve rôzne hodnoty, *uzol 0* rozvetvíme na dve vetvy, zodpovedajúce dvom disjunktným podmnožinám MPR. Prvá z nich (*uzol 1*) bude reprezentovať všetky prípustné riešenia, v ktorých je prijatá prvá zákazka (t.j. x_1 = 1) a druhá podmnožina (*uzol 2*) zase všetky riešenia, v ktorých je prvá zákazka zamietnutá (t.j. x_1 = 0). Matematické modely zodpovedajúce uzlom 1 a 2 zostavíme jednoducho dosadením príslušnej hodnoty premennej x_1 do modelu zodpovedajúcemu rozvetvovanému uzlu (v tomto prípade teda uzol 0).



Obr. 5 Grafické zobrazenie postupu riešenia úlohy metódou vetvenia a medzí

$$11 + 63x_2 + 9x_3 + 5x_4 + 4x_5 + 8x_6 \overset{!}{=} MAX$$

$$7x_2 + x_3 + x_4 + x_5 + 5x_6 \leq 14$$

$$70x_2 + 10x_3 + 5x_4 + 3x_5 + 2x_6 \leq 85$$

Uzol 1 *je charakterizovaný modelom:*

Horná medza pre tento uzol je $f_H(1) = 11 + 63 \cdot 1 + 9 \cdot 1 + 5 \cdot 1 + 4 \cdot 1 + 8 \cdot 1 = 100$.

Spodnú medzu stanovíme tak, že okrem už vybranej hodnoty x_1 = 1, ktorá charakterizuje množinu riešení v uzle 1, všetky ostatné premenné dáme nulové, čím zabezpečíme triviálne splnenie všetkých ohraničujúcich podmienok, t.j. $f_S(1) = f(1,0,0,0,0) = 11$. Táto hodnota je samozrejme lepšia ako aktuálna globálna hodnota $f_S = -\infty$, takže jej zodpovedajúce riešenie je aktuálne najlepšie nájdené riešenie a globálna hodnota f_S sa nám zmení z $-\infty$ na 11.

Analogicky vytvoríme ďalší uzol (s označením 2) vetvenia. Jeho model získame opäť z modelu rodičovského uzla (v tomto prípade uzol 0), dosadením hodnoty premennej x_1 = 0.

Uzol 2 *je charakterizovaný modelom:*

$$63x_2 + 9x_3 + 5x_4 + 4x_5 + 8x_6 \overset{!}{=} MAX$$
$$7x_2 + x_3 + x_4 + x_5 + 5x_6 \leq 15$$
$$70x_2 + 10x_3 + 5x_4 + 3x_5 + 2x_6 \leq 100$$

Horná medza pre tento uzol je $f_H(2) = 63 \cdot 1 + 9 \cdot 1 + 5 \cdot 1 + 4 \cdot 1 + 8 \cdot 1 = 89$. Spodná medza $f_S(2) = f(0,0,0,0,0) = 0$ nie je lepšia ako aktuálne platná spodná medza f_S = 11, takže nedochádza k jej zmene.

Krok orezávania v metóde vetvena a medzí vylúči z ďalšej analýzy všetky uzly, ktorých hodnota hornej medze je menšia alebo rovná ako aktuálnej hodnote spodnej medze. V našom prípade momentálne žiaden z dvoch uzlov, ktoré máme ešte nepreskúmané, nespĺňa uvedenú podmienku.

Môžme teda uzavrieť prvú iteráciu algoritmu a pokračovať krokom 2 pre vybraný uzol (resp. jemu zodpovedajúcu podmnožinu MPR). Je rozumné predpokladať, že najperspektívnejšia by mala byť tá podmnožina, ktorá má šancu na čo najlepšiu hodnotu kriteriálnej funkcie, t.j. ktorej hodnota hornej medze je maximálna spomedzi všetkých aktuálne otvorených podmnožín. V našom prípade máme momentálne otvorené dve podmnožiny (uzly 1 a 2), z ktorých lepšiu hodnotu hornej medze má uzol 1, takže pokračujeme jeho vetvením.

Vetvenie vykonáme podľa hodnoty druhej premennej v poradí, t.j. x_2. Vzniknú nám takto dva nové uzly, ktoré označíme poradovými číslami 3 (pre x_2 = 1), resp. 4 (pre x_2 = 0). Zodpovedajúce modely (odvodené z modelu uzla, ktorý rozvetvujeme, t.j. uzol 1) a vypočítané medze budú nasledovné.

$$74 + 9x_3 + 5x_4 + 4x_5 + 8x_6 \overset{!}{=} MAX$$
$$x_3 + x_4 + x_5 + 5x_6 \leq 7$$
$$10x_3 + 5x_4 + 3x_5 + 2x_6 \leq 15$$

Uzol 3 *je charakterizovaný modelom:*

Takže horná medza $f_H(3) = 74 + 9 \cdot 1 + 5 \cdot 1 + 4 \cdot 1 + 8 \cdot 1 = 100$ a spodná medza $f_S(3) = f(1,1,0,0,0,0) = 74$ čo je lepšie ako aktuálne najlepšie nájdené riešenie, takže dochádza k zmene globálnej hodnoty f_S na 74. Momentálne však táto hodnota nestačí na orezanie žiadneho z otvorených uzlov.

Uzol 4 *je charakterizovaný modelom:*

$$11 + 9x_3 + 5x_4 + 4x_5 + 8x_6 \overset{!}{=} MAX$$
$$x_3 + x_4 + x_5 + 5x_6 \leq 14$$
$$10x_3 + 5x_4 + 3x_5 + 2x_6 \leq 85$$

Takže horná medza $f_H(4) = 11 + 9 \cdot 1 + 5 \cdot 1 + 4 \cdot 1 + 8 \cdot 1 = 37$ čo je menej ako aktuálna spodná medza f_S = 74. Znamená to, že v množine riešení, ktoré

reprezentuje uzol 4 (t.j. takých, v ktorých je prijatá zákazka 1, ale odmientutá zákazka 2) určite nemôže byť riešenie lepšie ako to, ktoré sme už našli, t.j. aktuálna spodná medza $f_S(3) = f(1,1,0,0,0,0) = 74$. Uzol 4 môžeme preto orezať a v ďalšom výpočte ho už neuvažujeme.

Ako kandidáti na rozvetvenie sú aktuálne k dispozícii uzly 2 a 3. Nakoľko horná medza uzla 3 je vyššia ako horná medza uzla 2, vyberieme na rozvetvenie v ďalšom kroku práve uzol 3. Tento opäť rozvetvíme na dva nové uzly, zodpovedajúce podmnožinám MPR podľa hodnoty ďalšej premennej v poradí, teda x_3. Výsledkom budú uzly 5 a 6 s nasledovnými zodpovedajúcimi matematickými modelmi.

Uzol 5 *je charakterizovaný modelom:*

$$83 + 5x_4 + 4x_5 + 8x_6 \overset{!}{=} MAX$$
$$x_4 + x_5 + 5x_6 \leq 6$$
$$5x_4 + 3x_5 + 2x_6 \leq 5$$

Jemu zodpovedajúca horná medza bude $f_H(5) = 83 + 5 \cdot 1 + 4 \cdot 1 + 8 \cdot 1 = 100$ a spodná medza $f_S(5) = f(1,1,1,0,0,0) = 83$, čo je lepšie ako aktuálne najlepšie nájdené riešenie, takže dochádza k zmene globálnej hodnoty f_S na 83. Momentálne však táto hodnota nestačí na orezanie žiadneho z otvorených uzlov (t.j. uzla 2).

Uzol 6 *je charakterizovaný modelom:*

$$74 + 5x_4 + 4x_5 + 8x_6 \overset{!}{=} MAX$$
$$x_4 + x_5 + 5x_6 \leq 7$$
$$5x_4 + 3x_5 + 2x_6 \leq 15$$

Jemu zodpovedajúca horná medza bude $f_H(6) = 74 + 5 \cdot 1 + 4 \cdot 1 + 8 \cdot 1 = 91$. Spodná medza $f_S(6) = f(1,1,0,0,0,0) = 74$ nie je lepšia ako aktuálne platná spodná medza f_S = 83, takže nedochádza k jej zmene. Ani orezanie tohto uzla nie je možné vzhľadom na to, že jeho horná medza je vyššia ako aktuálna spodná medza, takže v tomto uzle sa môže skrývať lepšie riešenie ako to, ktoré sme už našli.

Pre rozvetvenie opäť vyberieme uzol s najvyššou hornou medzou, čo je v tomto prípade uzol 5. Jeho rozvetvením podľa hodnoty nasledujúcej premennej x_4 vzniknú nové dva uzly 7 (pre x_4 = 1) a 8 (pre x_4 = 0), ktorých modely sú odvodené z modelu východzieho uzla 5.

Uzol 7 *je charakterizovaný modelom:*

$$88 + 5x_4 + 4x_5 + 8x_6 \overset{!}{=} MAX$$
$$x_5 + 5x_6 \leq 5$$
$$3x_5 + 2x_6 \leq 0$$

Jemu zodpovedajúca horná medza bude $f_H(7) = 88 + 4 \cdot 1 + 8 \cdot 1 = 100$ a spodná

medza $f_S(7) = f(1,1,1,1,0,0) = 88$, čo je lepšie ako aktuálne najlepšie nájdené riešenie, takže dochádza k zmene globálnej hodnoty f_S na 88. Momentálne však táto hodnota nestačí na orezanie žiadneho z otvorených uzlov (t.j. uzlov 2 a 6).

Uzol 8 *je charakterizovaný modelom:*

$$83 + 4x_5 + 8x_6 \overset{!}{=} MAX$$
$$x_5 + 5x_6 \leq 6$$
$$3x_5 + 2x_6 \leq 5$$

Jemu zodpovedajúca horná medza bude $f_H(8) = 83 + 4 \cdot 1 + 8 \cdot 1 = 95$. Spodná medza $f_S(8) = f(1,1,1,0,0,0) = 83$ nie je lepšia ako aktuálne platná spodná medza f_S = 88, takže nedochádza k jej zmene. Ani orezanie tohto uzla nie je možné vzhľadom na to, že jeho horná medza je vyššia ako aktuálna spodná medza, takže v tomto uzle sa môže skrývať lepšie riešenie ako to, ktoré sme už našli.

Pre rozvetvenie opäť vyberieme uzol s najvyššou hornou medzou, čo je v tomto prípade uzol 7. Jeho rozvetvením podľa hodnoty nasledujúcej premennej x_5 vzniknú nové dva uzly 9 (pre x_5 = 1) a 10 (pre x_5 = 0), ktorých modely sú odvodené z modelu východzieho uzla 7.

Uzol 9 *je charakterizovaný modelom:*

$$93 + 4x_5 + 8x_6 \overset{!}{=} MAX$$
$$5x_6 \leq 4$$
$$2x_6 \leq -3$$

Je jasné, že druhá ohraničujúca podmienka nemôže byť splnené prežiadnu hodnotu z definičného oboru premennej x_6, preto podmnožina zodpovedajúca uzlu 9 je prádzna, neobsahuje žiadne prípustné riešenie. Tento uzol preto preškrtneme krížikom a pokračujeme charakterizáciou uzla 10.

Uzol 10 *je charakterizovaný modelom:*

$$88 + 8x_6 \overset{!}{=} MAX$$
$$5x_6 \leq 5$$
$$2x_6 \leq 0$$

Jemu zodpovedajúca horná medza bude $f_H(10) = 88 + 8 \cdot 1 = 96$ a spodná medza $f_S(10) = f(1,1,1,1,0,0) = 88$, čo nie je lepšie ako aktuálne najlepšie nájdené riešenie, takže nedochádza k zmene globálnej hodnoty f_S a nie je preto možné ani žiadne orezávanie.

Pre rozvetvenie máme momentálne k dispozícii ešte otvorené uzly 2, 6, 8 a 10, z ktorých najvyššiu hornú medzu má uzol 10, takže vyberieme na vetvenie práve tento uzol. Rozvetvením podľa hodnoty poslednej nasledujúcej premennej x_6 nám vzniknú nové dva uzly, ktoré už budú listové

vo vytváranom strome prehľadávania, nakoľko všetky premenné v nich už majú priradené hodnoty. Zároveň to znamená, že podmnožiny MPR na úrovni listových uzlov môžu byť maximálne jednoprvkové, t.j. reprezentujú jedno konkrétne riešenie, alebo sú prázdne a neobsahujú prípustné riešenie.

V našom prípade nám vzniknú vetvením uzla 10 dva listové uzly 11 (pre x_6 = 1) a 12 (pre x_6 = 0), ktorých modely sú odvodené z modelu východzieho uzla 10.

Listový uzol 11 *je charakterizovaný modelom:*

$$96 \overset{!}{=} MAX$$
$$5 \le 5$$
$$2 \le 0$$

Je zjavné že druhá ohraničujúca podmienka je porušená, takže tento listový uzol je prázdny, neobsahuje žiadne prípustné riešenie a preto ho preškrtneme krížikom.

Listový uzol 12 *je charakterizovaný modelom:*

$$88 \overset{!}{=} MAX$$
$$0 \le 5$$
$$0 \le 0$$

Ide vlastne o riešenie, ktoré sme našli už v nadradenom uzle 7 a predstavuje aktuálnu spodnú medzu a najlepšie doteraz nájdené riešenie, t.j.

$f_S(10) = f_S(7) = f(1,1,1,1,0,0) = 88$. Globálne stanovená spodná medza sa teda nemení a nie je možné preto ani žiadne orezávanie.

Stále však ešte máme niekoľko otvorených uzlov, a síce uzly 2, 6 a 8. Z nich najvyššiu hodnotu hornej medze má uzol 8, preto ho vyberieme na vetvenie. Jeho vetvením podľa hodnoty v ňom nasledujúcej premennej x_5 nám vzniknú nové dva uzly, ktoré označíme poradovými číslami 13 a 14. Ich modely budú odvodené z modelu uzla 8 dosadením príslušnej hodnoty premennej x_5, konkrétne x_5 = 1 pre uzol 13 a x_5 = 0 pre uzol 14.

Uzol 13 *je charakterizovaný modelom:*

$$87 + 8x_6 \overset{!}{=} MAX$$
$$5x_6 \le 5$$
$$2x_6 \le 2$$

Jemu zodpovedajúca horná medza bude $f_H(13) = 87 + 8 \cdot 1 = 95$. Spodná medza $f_S(13) = f(1,1,1,0,1,0) = 87$ nie je lepšia ako aktuálne platná spodná medza f_S = 88, takže nedochádza k jej zmene a tým pádom ani k žiadnemu orezávaniu.

Uzol 14 *je charakterizovaný modelom:*

$$83 + 8x_6 \overset{!}{=} MAX$$
$$5x_6 \le 6$$
$$2x_6 \le 5$$

Jemu zodpovedajúca horná medza bude $f_H(14) = 83 + 8 \cdot 1 = 91$. Spodná medza $f_S(14) = f(1,1,1,0,0,0) = 83$ nie je lepšia ako aktuálne platná spodná medza f_S = 88. Ani orezať daný uzol nie je možné, nakoľko jeho horná medza je stále ešte vyššia ako aktuálna spodná medza.

Vyberáme preto opäť najperspektívnejší uzol na vetvenie, tentokrát z uzlov 2, 6, 13, a 14. Z nich najvyššiu hodnotu hornej medze má uzol 13, takže rozvetvíme práve tento na dva opäť už listové uzly, podľa hodnoty poslednej nasledujúcej premennej x_6 na uzol 15 pre x_6 = 1 a uzol 16 pre x_6 = 0.

Listový uzol 15 *je charakterizovaný modelom:*

$$95 \overset{!}{=} MAX$$

$$5 \leq 5$$

$$2 \leq 2$$

Nakoľko ide o listový uzol, nachádza sa v ňom práve jedno prípustné riešenie, keďže ohraničujúce podmienky sú splnené. Toto riešenie zodpovedá hodnote kriteriálnej funkcie $f_S(15) = f(1,1,1,0,1,1) = 95$ čo je najlepšie doteraz nájdené riešenie, takže mení globálnu hodnotu spodnej medze na f_S = 95.

Táto hodnota je vyššia ako horná medza otvorených uzlov 2, 6 a 14, takže tieto môžme orezať. Ostáva nám uzol 16, ktorý ale nemôže obsahovať lepšie riešenie, nakoľko sa líši od listového uzla 15 zamietnutím poslednej zákazky, čo nutne zníži výsledký zisk (hodnotu kriteriálnej funkcie). Algoritmus teda končí s tým, že najlepšie prípustné riešenie dosahuje hodnotu zisku 95 a spočíva v prijatí všetkých zákaziek okrem štvrtej.

2.3 Úlohy s ohraničeniami

Tak úlohy lineárneho ako aj celočíselného programovania patria do všeobecnejšej kategórie úloh, ktoré sa označujú úlohy s ohraničeniami (angl. constraint satisfaction problems). Tieto úlohy sú charakterizované trojicou množín:

1. Množina premenných
2. Množina domén premenných (ich definičný obor)
3. Množina ohraničení medzi premennými

Riešiť úlohu s ohraničeniami môže znamenať

- dokázať existenciu, resp. neexistenciu jej riešenia,
- nájsť jedno jej riešenie,
- nájsť všetky riešenia,
- alebo nájsť optimálne riešenie (ak je zadaná naviac aj kriteriálna funkcia).

Podrobne je takýmto úlohám a ich riešeniu pomocou logického programovania ohraničení venovaná kniha [Mach, Paralič 2000], takže

v tomto učebnom texte sa jej podrobnejšie nebudem venovať.

Okrem toho je tento prístup popísaný v kontexte jeho použitia pri riešení úloh rozvrhovania v časti 6.4.10 kapitoly o rozvrhovaní v tomto učebnom texte.

3 Alokačné úlohy

Existuje mnoho úloh tohoto typu. Dôvod, prečo je tento druh úloh zaradený na začiatok celého reťazca výrobnej logistiky (viď. Obr. 1 v časti 1.2) je ten, že jednou z alokačných úloh je rozhodnutie, ktoré je potrebné prijať hneď na začiatku – o umiestnení výroby, kde spravidla existuje viacero alternatív. Druhou takouto úlohou je napr. rozhodnutie o umiestnení distribučných centier a pod.

Vo všeobecnosti budeme alokačné úlohy deliť na dve skupiny, alokácia do jedného miesta (podrobnejšie v nasledujúcej časti 3.1) a alokácia do viacerých miest (podrobnejšie v časti 3.2).

3.1 Alokácia do jedného miesta

Z úloh, ktoré sa snažia nájsť najlepšie možné umiestnenie jedného objektu, si uvedieme dva typy situácií. Prvý typ je prípad, keď nie je možné presne vyčísliť náklady, spravidla pri úlohe kam umiestniť výrobnú prevádzku (podrobnejšie viď. nasledujúca časť 3.1.1).

Druhý typ je potom charakteristický presne definovanými nákladmi a intenzitou väzieb medzi objektom, ktorého optimálne umistnenie hľadáme, a ostatnými objektami, s ktorými bude mať väzbu (podrobnejšie viď. časť 3.1.2).

3.1.1 Alokácia výrobného procesu do jedného miesta

Táto časť je spracovaná podľa [Malindžák 1996]. Pre úlohu alokácie výrobného procesu do jedného miesta sú typické nasledovné charakteristiky:

- sú známe lokality, kde je možné umiestniť výrobný proces a treba vybrať najvhodnejšiu z nich,
- je ťažké vyčísliť presné náklady spojené s umiestnením výrobného procesu do jednotlivých alternatívnych lokalít,
- nie sú presne známi dodávatelia a odberatelia,
- existuje veľa faktorov, ktoré je ťažko ohodnotiť, ale je možné vyjadriť ich závažnosť voči ostatným faktorom a porovnať ich hodnoty pre jednotlivé lokality.

V takomto prípade sa využíva tzv. ***pomerovo-indexová metóda***. Postup je nasledovný.

---

1. Pre vybrané lokality (L = 1 ... n) a daný výrobný proces vyberáme rozhodujúce faktory(F_1 ... F_m). Každému faktoru F_i prisúdime váhu w_i najlepšie tak, aby suma všetkých váh bola 1. F_i (i = 1 ... m): w_i , $\sum_{i=1}^{m} w_i = 1$

2. Pre hodnotenie jednotlivých faktorov F_i zvolíme interval hodnotenia (tzv. kardinálnu mieru) HF_i, ktorý bude mať hornú hranicu KH_i a dolnú hranicu KD_i, t.j. $HF_i \in \langle KD_i, KH_i \rangle$

3. Experti stanovia hodnotenie HF_i^L (L = 1 ... n, i = 1 ... m) pre všetky lokality L a pre všetky faktory F_i.

4. Výsledné hodnotenie danej lokality L je dané váženým súčtom: $C^L = \sum_{i=1}^{m} w_i HF_i^L$

5. Ako najlepšia bude vybraná tá lokalita, pre ktorú je hodnota C^L maximálna, t.j. $L = \max C^L$

Príklad

Úlohou je vybrať najvhodnejšiu lokalitu pre umiestnenie výroby drevených hračiek z troch vytipovaných lokalít (Spišská Nová Ves – SNV, Rožňava – RV, Svidník – Sk).

Zadané faktory, ich váhy, hodnotenia pre jednotlivé lokality ako aj výpočet jednotlivých hodnôt podľa pomerovo-indexovej metódy je zhrnutý na Obr. 6.

cislo	Faktory	W_i	SNV $[L_1]$	$W_i HF_i^1$	RV $[L_2]$	$W_i HF_i^2$	Sk $[L_3]$	$W_i HF_i^3$
1	Suroviny	0.13	8	1.04	6	0.78	7	0.91
2	Doprava	0.09	8	0.72	4	0.36	6	0.54
3	Energia	0.09	4	0.36	4	0.36	2	0.18
4	Voda	0.06	8	0.48	5	0.30	9	0.54
5	Financie	0.18	7	1.26	2	0.36	6	1.08
6	Odbyt	0.20	5	1.00	2	0.40	4	0.80
7	Spoje	0.11	7	1.77	7	0.77	3	0.33
8	Prac. sily	0.13	5	0.65	5	0.65	5	0.65
		$\sum_{i=1}^{8} W_i = 1$		$c^1 = 7.05$		$c^2 = 3.98$		$c^3 = 4.03$

Obr. 6 Zadanie a riešenie príkladu úlohy alokácie výrobného procesu do jedného miesta, prevzaté z [Malindžák 1996]

3.1.2 Optimálne umiestnenie distribučného centra

V tomto prípade ide o exaktne zadanú úlohu, ktorá je predmetom skúmania aj v rámci operačného výskumu [Dudorkin 1997].

Úloha je definovaná nasledovne. V rovine existuje m objektov (odberateľov) (P_1, ..., P_m) so súradnicami (a_1, b_1), … (a_m, b_m). Treba nájsť súradnice pre umiestnenie nového objektu (distribučného centra) $\bar{x}$ = (x, y) tak, aby celkové náklady na realizáciu väzieb medzi existujúcimi objektmi a novým objektom boli minimálne. Intenzitu väzby medzi objektmi P_i a novým objektom $\bar{x}$ vyjadrujú koeficienty w_i (i = 1, ..., m).

Kritériom je prirodzene minimalizácia nákladov. Tie sú dané súčtom prepravných nákladov od distribučného centra ku všetkým jeho odberateľom, pričom náklady na pokrytie spotreby odberateľa P_i sú dané násobkom intenzity väzby a vzdialenosti medzi distribučným centrom a daným odberateľom, t.j. $w_i d(\bar{x}, P_i)$.

Takže celkové náklady budú:

$$f(\bar{x}) = \sum_{i=1}^{m} w_i d(\bar{x}, P_i)$$.

Vzdialenosť môže byť ale počítaná rôzne. V praktických aplikáciách sa môže vyskytnúť jeden z nasledovných štyroch spôsobov výpočtu vzdialenosti.

a. ***Euklidovská*** $\sqrt{(x-a_i)^2 + (y-b_i)^2}$

b. ***Kvadrát euklidovskej*** $(x-a_i)^2 + (y-b_i)^2$

c. ***Rektilineárna*** $\min_{x,y}\left[w_i\left(|x-a_i| + |y-b_i|\right)\right]$

d. ***Minimalizácia najvzdialenejšieho bodu***

$$\min\left[\max\sqrt{(x-a_i)^2 + (y-b_i)^2}\right]$$

Pre každý typ vzdialenosti je iný postup výpočtu optimálneho umiestnenia nového objektu (distribučného centra).

a. Pre **euklidovskú vzdialenosť** sa používa numerické riešenie (tzv. hyperbolická aproximácia). Hľadáme extrém funkcie dvoch premenných (súradnica x a súradnica y pre umiestnenie nového objektu – distribučného centra), preto derivujeme funkciu nákladov

$$f(\bar{x}) = \sum_{i=1}^{m} w_i \sqrt{(x-a_i)^2 + (y-b_i)^2}$$

parciálne a jednotlivé parciálne derivácie položíme rovné nule, t.j.:

$$\frac{\partial f(\bar{x})}{\partial x} = \sum_{i=1}^{m} \frac{w_i(x-a_i)}{\sqrt{(x-a_i)^2 + (y-b_i)^2}} \stackrel{!}{=} 0 \text{, resp. } \frac{\partial f(\bar{x})}{\partial y} = \sum_{i=1}^{m} \frac{w_i(y-b_i)}{\sqrt{(x-a_i)^2 + (y-b_i)^2}} \stackrel{!}{=} 0$$

Po úprave a zavedení substitúcie $g_i(x,y)$ dostávame postupne pre x-ovú súradnicu:

$$x\sum_{i=1}^{m} \frac{w_i}{\sqrt{(x-a_i)^2 + (y-b_i)^2}} = \sum_{i=1}^{m} \frac{w_i a_i}{\sqrt{(x-a_i)^2 + (y-b_i)^2}}$$

$$g_i(x,y) = \frac{w_i}{\sqrt{(x-a_i)^2 + (y-b_i)^2 + \xi}}$$

$$x\sum_{i=1}^{m} g_i = \sum_{i=1}^{m} a_i g_i$$

$$x^{(k)} = \frac{\sum_{i=1}^{m} a_i g_i\left(x^{(k-1)}, y^{(k-1)}\right)}{\sum_{i=1}^{m} g_i\left(x^{(k-1)}, y^{(k-1)}\right)}$$

Analogicky pre pre y-ovú súradnicu dostávame:

$$y \cdot \sum_{i=1}^{m} \frac{w_i}{\sqrt{(x-a_i)^2+(y-b_i)^2}} = \sum_{i=1}^{m} \frac{w_i b_i}{\sqrt{(x-a_i)^2+(y-b_i)^2}}$$

$$g_i(x,y) = \frac{w_i}{\sqrt{(x-a_i)^2+(y-b_i)^2+\xi}}$$

$$y\sum_{i=1}^{m} g_i = \sum_{i=1}^{m} b_i g_i$$

$$y^{(k)} = \frac{\sum_{i=1}^{m} b_i g_i\left(x^{(k-1)}, y^{(k-1)}\right)}{\sum_{i=1}^{m} g_i\left(x^{(k-1)}, y^{(k-1)}\right)}$$

Výsledkom sú iteračné vzorce pre výpočet súradníc optimálneho umiestnenia distribučného centra $x^{(k)}$ a $y^{(k)}$. Na začiatku stanovíme hodnoty ($x^{(0)}$ a $y^{(0)}$) pre ťažisko, t.j. $x^{(0)} = \frac{\sum_{i=1}^{m} a_i w_i}{\sum_{i=1}^{m} w_i}$, resp. $y^{(0)} = \frac{\sum_{i=1}^{m} b_i w_i}{\sum_{i=1}^{m} w_i}$

a postupne v každej ďalšej iterácii konverguje riešenie k optimu. Po dosiahnutí požadovanej presnosti (napríklad že hodnota kriteriálnej funkcie na druhom ráde za desatinnou čiarkou sa už nemení) výpočet ukončíme a aktuálne hodnoty $x^{(k)}$ a $y^{(k)}$ určujú doporučené umiestnenie distribučného centra.

Príklad

Je potrebné nájsť optimálne umiestnenie trafostanice pre 4 stanice s danými súradnicami: A[2,6], B[6,7], C[7,4], D[5,2], káblom s mernými ročnými nákladmi 3 PJ/km (PJ znamená skratku pre „peňažná jednotka“). Nová stanica bude napájaná káblom s ročnými nákladmi 5 PJ/km z existujúcej trafostanice E[1,1].

Zadané súradnice jednotlivých odberateľských miest ako aj napájacej trafostanice spolu s mernými ročnými nákladmi (intenzity väzieb) sú zhrnuté v ľavej tabuľke na Obr. 7.

i	Miesto	a_i	b_i	w_i
1	A	2	6	3
2	B	6	7	3
3	C	7	4	3
4	D	5	2	3
5	E	1	1	5

i	$x^{(i)}$	$y^{(i)}$	$f(x^{(i)}, y^{(i)})$
0	3.82	3.65	55.935
1	3.98	3.53	55.772
2	4.06	3.47	55.730
3	4.10	3.44	55.719
4	4.12	3.42	55.716

Obr. 7 Zadanie a riešenie príkladu úlohy optimálneho umiestnenia trafostanice (alokácia do jedného miesta, euklidovská vzdialenosť) [Dudorkin 1997]

Riešenie:

Vyjdeme z počiatočných hodnôt súradníc $x^{(0)}$ a $y^{(0)}$ pre ťažisko a vypočítame zodpovedajúcu hodnotu kriteriálnej funkcie $f(x^{(0)}, y^{(0)})$. Potom vypočítame substitučné koeficienty $g_i(x^{(0)}, y^{(0)})$ a dosadíme ich do iteračných vzorcov pre výpočet $x^{(1)}$ a $y^{(1)}$.

$$x^{(0)} = \frac{3.(2+6+7+5)+5.1}{3+3+3+3+5} = \frac{65}{17} = 3{,}82$$

$$y^{(0)} = \frac{3.(6+7+4+2)+5.1}{3+3+3+3+5} = \frac{62}{17} = 3{,}65$$

$$f\left(x^{(0)}, y^{(0)}\right) = 3.\sqrt{(3{,}82-2)^2+(3{,}65-6)^2} + \ldots + 5.\sqrt{(3{,}82-1)^2+(3{,}65-1)^2} = 55{,}935$$

$$g_1\left(x^{(0)}, y^{(0)}\right) = \frac{3}{.\sqrt{(3{,}82-2)^2+(3{,}65-6)^2+0{,}001}} = 1{,}0092$$

...

$$g_1\left(x^{(0)}, y^{(0)}\right) = \frac{5}{.\sqrt{(3{,}82-1)^2+(3{,}65-1)^2+\xi}} =$$

Opäť vypočítame hodnotu kriteriálnej funkcie pre nové umiestnenie distribučného centra $f(x^{(1)}, y^{(1)})$ a celý postup iteratívne opakujeme až do chvíle, kým zmena hodnoty kriteriálnej funkcie v dvoch po sebe nasledujúcich iteráciách klesne pod jednu stotinu PJ (viď. pravá tabuľka na Obr. 7).

b. Ak sa vzdialenosť meria ako **kvadrát euklidovskej vzdialenosti**, t.j. $d(\bar{x}, P_i) = (x-a_i)^2 + (y-b_i)^2$, potom je optimálnym umiestnením distribučného centra jeho ťažisko,t.j.

$$x^{(0)} = \sum_{i=1}^{m} \frac{a_i w_i}{w_i}, \quad y^{(0)} = \sum_{i=1}^{m} \frac{b_i w_i}{w_i}$$

c. V prípade **rektilineárnej vzdialenosti** sa používa na výpočet optimálneho umiestnenia distribučného centra tzv. mediánové umiestenie. Optimálne hodnoty pre súradnicu x aj y totiž musia ležať v x-ovej, resp. **y**-ovej súradnici niektorého zo vstupných objektov (pre každú súradnicu to samozrejme môže byť iný objekt).

V tomto prípade je potrebné najprv jednotlivé objekty usporiadať vzostupne podľa ich súradnice x (resp. y), a potom vypočítať jednotlivé čiastkové súčty váh w_i príslušných týmto objektom, t.j. pre x-ovú súradnicu:

$$a_1 \le a_2 \le ... \le a_m$$

$$s_k = \sum_{i=1}^{k} w_i$$

$$s_m = \frac{1}{2} \sum_{i=1}^{k} w_i$$

$$s_1 \le s_2 \le ... \le s_m$$

$$s_{k-1} \le s_m \le s_k$$

Analogicky pre y-ovú súradnicu:

$$b_1 \le b_2 \le ... \le b_m$$

$$s_k = \sum_{i=1}^{k} w_i$$

$$s_m = \frac{1}{2} \sum_{i=1}^{k} w_i$$

$$s_1 \le s_2 \le ... \le s_m$$

$$s_{k-1} \le s_m \le s_k$$

Príklad

Použijeme tie isté vstupné údaje ako v príklade pre prípad euklidovskej vzdialenosti vyššie (viď. ľavá tabuľka na Obr. 7). Pre x-ovú súradnicu dostávame túto postupnosť: $a_5 \le a_1 \le a_4 \le a_2 \le a_3$, takže tomu zodpovedajúca postupnosť čiastkových súčtov váh bude:

$s_5 = 5$

$s_1 = 5 + 3 = 8$

$s_4 = 5 + 3 + 3 = 11$

$s_2 = 5 + 3 + 3 + 3 = 14$

$s_3 = 5 + 3 + 3 + 3 + 3 = 17$

Medián je $s_m = \frac{1}{2} \cdot 17 = 8{,}5$ a teda jemu zodpovedajúce mediánové umiestnenie je $a_4 = 5$ (nakoľko 8,5 sa nachádza medzi s_1 a s_4).

Pre y-ovú súradnicu dostávame túto postupnosť: $b_5 \le b_4 \le b_3 \le b_1 \le b_2$, takže tomu zodpovedajúca postupnosť čiastkových súčtov váh bude:

$s_5 = 5$

$s_4 = 5 + 3 = 8$

$s_3 = 5 + 3 + 3 = 11$

$s_1 = 5 + 3 + 3 + 3 = 14$

$s_2 = 5 + 3 + 3 + 3 + 3 = 17$

Medián je opäť $s_m = \frac{1}{2} \cdot 17 = 8{,}5$ a tomu zodpovedajúce mediánové umiestnenie je $b_3 = 4$ (nakoľko 8,5 sa nachádza medzi s_4 a s_3).

d. V prípade **minimalizácie vzdialenosti najvzdialenejšieho objektu** je optimálnym umiestnením distribučného centra stred kružnice s minimálnym polomerom opísanej tak, že v nej ležia všetky vstupné objekty (všetci odberatelia).

3.2 Alokácia do viacerých miest

V prípade alokácie do viacerých miest potrebujeme umiestniť viacero objektov. Táto úloha má viacero typov, ktoré sa líšia tým, akým spôsobom vznikajú náklady pri umiestňovaní a prevádzke nových objektov.

3.2.1 Priradzovací problém

Majme n objektov, ktoré je potrebné umiestniť do n miest s minimálnymi nákladmi. Ak poznáme náklady c_{ij} ($i = 1 \ldots n,\ j = 1 \ldots n$) pre lokalizáciu i-teho objektu do j-teho miesta, takúto úlohu nazývame priradzovací problém (základná verzia).

Základnú verziu priradzovacieho problému možno riešiť pomocou celočíselného programovania. Jednoduchý bivalentný model vyzerá nasledovne.

Pre každý umiestňovaný objekt zavedieme jednu bivalentnú premennú, t.j. $x_{ij} \in \{0,1\}$, pričom x_{ij} = 1 ak i-ty objekt bude umiestnený do j-teho miesta a x_{ij} = 0 ak i-ty objekt nebude umiestnený do j-teho miesta.

Kriteriálna funkcia vyjadrujúca úhrnné náklady pre dané rozmiestnenie objektov je nasledovná.

$$f(\bar{x}) = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \overset{!}{=} MIN$$

Ohraničujúce podmienky budú dvoch typov. Prvá skupina ohraničujúcich podmienok vyjadruje pre každý objekt tú skutočnosť, že môže byť umiestnený do práve jedného miesta, čo možno zapísať nasledovne:

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j = 1..n$$

Druhá skupina ohraničení je zase zameraná na jednotlivé miesta. Pre každé miesto musí platiť, že je do neho umiestnený práve jeden objekt, t.j.:

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i = 1..n$$

3.2.2 Priradzovací problém (väzby na existujúce objekty)

Majme v priestore už p existujúcich objektov, pričom chceme do tohto priestoru rozmiestniť n nových objektov a k dispozícii máme n miest. V prípade že existujú väzby medzi novými a existujúcimi objektmi, dostávame opäť priradzovací problém, ale náklady c_{ij} si v tomto prípade musíme vypočítať.

Obvykle je známa matica vzdialeností medzi existujúcimi objektmi a novými miestami $\overline{D} = \left[d_{ij}\right]_n^P$ a matica prepravných sadzieb $\overline{W} = \left[w_{ij}\right]_n^P$, ktoré definujú intenzitu väzbieb medzi starými a novými objektmi. Potom maticu nákladov možno vypočítať nasledovne: $\overline{C} = \overline{W}\,\overline{D} = \left[c_{ij}\right]_n^n$.

Výsledný bivaletný model je potom úplne rovnaký ako pri základnej verzii priradzovacieho problému (viď. predchádzajúca časť 3.2.1).

Príklad

V tabuľke zobrazenej na Obr. 8 sú uvedené vzdialenosti v [m] medzi existujúcimi strojmi P, O, R a novými strojmi A, B, C. Z priestorových dôvodov nemožno premiestniť stroj B do miesta H. Nájdite optimálne rozmiestnenie nových objektov do nových miest E, F, G, H.

	Existujuce stroje $[ks]$	P	O	R
Nove stroje $[ks]$	A	5	4	2
	B	0	4	3
	C	4	3	2
	D	0	0	0
Mozne miesta $[m]$	E	1	3	4
	F	4	4	3
	G	5	3	5
	H	6	4	2

Obr. 8 Zadanie príkladu priradzovacieho problému podľa [Dudorkin 1997]

Riešenie:

Najprv je potrebné vypočítať náklady spojené s umiestnením jednotlivých objektov do jednotlivých alternatívnych miest, t.j. maticu nákladov $\overline{C}$. Problém s menším počtom strojov než je dostupných miest sa rieši pridaním fiktívnych strojov s nulovými intenzitami väzieb tak, aby celkový počet strojov bol rovnaký ako počet voľných miest. V našom prípade doplníme jeden fiktívny stroj D s nulovými intenzitami väzieb w_{Dx} na všetky existujúce stroje.

$$\overline{C} = \overline{W}\,\overline{D} = \begin{bmatrix} 5 & 4 & 2 \\ 0 & 4 & 3 \\ 4 & 3 & 2 \\ 0 & 0 & 0 \end{bmatrix} . \begin{bmatrix} 1 & 4 & 5 & 6 \\ 3 & 2 & 3 & 4 \\ 4 & 3 & 5 & 2 \end{bmatrix} = \begin{bmatrix} 25 & 34 & 47 & 50 \\ 24 & 17 & 27 & 1000 \\ 21 & 28 & 39 & 40 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Hodnotu nákladov c_{BH} prepíšeme na vysokú hodnotu (napr. 1000), ktorá zabráni priradeniu stroje B do miesta H, nakoľko zo zadania úlohy vyplýva, že sa tam nezmestí.

Potom môžme sformulovať bivaletný model optimalizačnej úlohy na riešenie priradzovacieho problému.

$$f(\bar{x}) = \sum_{i=1}^{4}\sum_{j=1}^{4} c_{ij}\, x_{ij} \stackrel{!}{=} MIN$$

$$\sum_{i=1}^{4} x_{ij} = 1 \qquad \forall j = 1..4$$

$$\sum_{j=1}^{4} x_{ij} = 1 \qquad \forall i = 1..4$$

Tento bivaletný problém možno vyriešiť napr. metódou vetvenia a medzí alebo inou metódou na riešenie úloh celočíselného programovania.

3.2.3 Kvadratický priradzovací problém

V tejto verzii pripradzovacieho problému sú definované väzby medzi novo umiestňovanými objektami navzájom medzi sebou.

Máme umiestniť n nových objektov do n miest. Medzi novými objektmi existujú vzájomné väzby. Je známa matica vzdialeností $\overline{D} = \left[d_{ij}\right]_n^n$, kde d_{ij} je vzdialenosť medzi i-tym a j-tym miestom. Takisto je známa matica prepravných sadzieb $\overline{W} = \left[w_{ij}\right]_n^n$, kde w_{ij} je intenzita väzby medzi i-tym a j-tym novým objektom.

Každé prípustné riešenie možno vyjadriť ako permutáciu $\overline{P} = (p(1), p(2),..., p(n))$, kde $p(i)=k$ znamená, že i-ty objekt bude umiestnený do miesta k.

Náklady pre rozmiestnenie objektov podľa zadanej permutácie sú:

$$f(\bar{P}) = \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\, d_{P(i)P(j)}$$

Na riešenie takto definovanej úlohy možno použiť metódu vetvenia a medzí, alebo heuristickú metódu CRAFT, ktorej popis nasleduje.

---

1. Z východzej (náhodnej) permutácie sa vytvorí $\binom{n}{2}$ nových permutácií výmenami všetkých dvojíc objektov vo východzej permutácii.
2. Pre každú permutáciu sa vypočíta hodnota kriteriálnej funkcie.
3. Vyberie sa riešenie s najlepšiou hodnotou kriteriálnej funkcie toto sa stane východzou permutáciou pre nasledujúcu iteráciu algoritmu.
4. Celý postup sa opakuje dovtedy, kým sa zlepšuje kriteriálna funkcia z jednej iterácie na druhú.

Príklad

Štyri nové stroje môžu byť umiestené do miest A, B, C, D. Vzdialenosti medzi

novými miestami sú uvedené v matici $\overline{D}$, denné počty prepravovaných paliet medzi dvojicami nových strojov sú v matici $\overline{W}$. Jednotkové prepravné náklady sú rovnaké. Nájdite čo najlepšie možné rozmietnenie nových strojov.

$$\overline{D} = \begin{bmatrix} 0 & 4 & 5 & 6 \\ 4 & 0 & 3 & 6 \\ 5 & 3 & 0 & 3 \\ 6 & 6 & 3 & 0 \end{bmatrix} \qquad \overline{W} = \begin{bmatrix} 0 & 4 & 1 & 3 \\ 4 & 0 & 2 & 0 \\ 1 & 2 & 0 & 7 \\ 3 & 0 & 7 & 0 \end{bmatrix}$$

Riešenie:

Vyjdeme z náhodného rozmiestnenia objektov reprezentovaného napr. permutáciou $\overline{P} = (3,1,4,2)$. Pre takéto rozmiestnenie objektov je hodnota kriteriálnej funkcie nasledovná:

$$f(\overline{P}) = w_{12}d_{CA} + w_{13}d_{CD} + w_{14}d_{CB} + w_{23}d_{AD} + w_{24}d_{AB} + w_{34}d_{DB} = 4.5 + 1.3 + 3.3 + 2.6 + 0.4 + 7.6 = 86$$

Všetkými možnými výmenami dvojíc objektov vytvoríme nové (susedné) permutácie a pre každú z nich vypočítame hodnotu kriteriálnej funkcie (viď. Obr. 9).

Stroj	Stroj	$f(\overline{p})$
1 2 3 4	1 2 3 4	
C A D B	A C D B	86
	D A C B	76
	B A D C	64
	C D A B	66
	C B D A	84
	C A B D	82

Obr. 9 Postup metódou CRAFT (1. iterácia) pri riešení príkladu pre kvadratický priradzovací problém [Dudorkin 1997].

Najlepšia hodnota kriteriálnej funkcie v 1. iterácii zodpovedá permutácii (2,1,4,3), s hodnotou 64. Preto táto permutácia sa stane východzou pre nasledujúcu, druhú iteráciu (viď. Obr. 10).

Stroj	Stroj	$f(\overline{p})$
1 2 3 4	1 2 3 4	
B A D C	A B D C	70
	D A B C	68
	C A D B	86
	B D A C	84
	B C D A	78
	B A C D	68

Obr. 10 Postup metódou CRAFT (2. iterácia) pri riešení príkladu pre kvadratický priradzovací problém [Dudorkin 1997].

Najlepšia hodnota kriteriálnej funkcie je po druhej iterácii 68, čo nie je lepšie ako hodnota predchádzajúcej permutácie, takže výpočet končí. Výpočet je

možné opakovať podľa potreby niekoľko krát pre ľubovoľné východzie permutácie.

3.2.4 Zovšeobecnený distribučný problém

Ide o najzložitejší variant alokačných úloh do viacerých miest. Úloha je zadaná nasledovne. Výrobca dodáva tovar n odberateľom a má k dispozícii konečný počet m miest pre zriadenie distribučného centra. Pre každé miesto sú určené fixné náklady f_i spojené so zriadením distribučného centra a prepravné náklady c_{ij} od i-teho distribučného centra j-temu odberateľovi. Úlohou je vybrať miesta pre zriadenie distribučných centier tak, aby celkové náklady (fixné aj prepravné) boli minimálne.

Tento problém možno riešiť viacerými spôsobmi. Vyberme po jednom z každého z troch základných prístupov pre riešenie úloh logistiky spomínaných v časti 1.3.

1. Operačný výskum reprezentovaný ***celočíselným programovaním***.
2. ***Heuristický prístup*** naprogramovaný klasickým procedurálnym prístupom (napr. v Pascale, alebo v jazyku C).
3. Z metód umelej inteligencie zoberme ***logické programovanie ohraničení***, ako deklaratívny programovací prístup na riešenie úloh s ohraničeniami.

Uveďme stručne charakteristiku, výhoy a nevýhody každého z uvedených troch prístupov.

1. Pri použití ***celočíselného programovania*** bude model vyzerať nasledovne. Potrebujeme dve skupiny premenných.

   Premenné y_i (i = 1, 2, ..., m) budú reprezentovať výber, resp. nevýber danej lokality pre umiestnenie distribučného centra, t.j. y_i = 1 ak i-te miesto bude vybrané pre zriadenie distribučného centra, ináč y_i = 0.

   Druhá skupina premenných bude určovať, ktoré distribučné centrá budú dodávať ktorým odberateľom. T.j. x_{ij} = {0, 1} (i = 1, 2, ..., m; j = 1, 2, ..., n) pričom x_{ij} = 1 ak i-te distribučné centrum bude dodávať j-temu odberateľovi, ináč x_{ij} = 0. Pritom predpokladáme, že každý odberateľ odoberá len od jedného distribučného centra.

   Celkové náklady budú dané súčtom fixných a prepravných náladov, t.j.

$$f\left(\bar{x},\bar{y}\right)=\sum_{i=1}^{m} f_i y_i+\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \overset{!}{=} MIN$$

   Čo sa týka ohraničujúcich podmienok, jedna skupina ohraničení zabezpečí to, že každý odberateľ bude uspokojený, t.j. bude odoberať od práve jedného distribučného centra:

$$\sum_{i=1}^{m} x_{ij}=1 \qquad \left(j=1,2,...,n\right)$$

   Druhá skupina ohraničení zabezpečí, že distribučné centrum bude môcť dodávať len v tom prípade, ak bude naozaj zriadené (zviazanie

premenných y_i a x_{ij}).

$$x_{ij} \le y_i \qquad (i = 1,2,...,m; \quad j = 1,2,...,n)$$

To znamená že budeme mať spolu $2^{n+nm}=2^{n(1+m)}$ ohraničení. Pre úlohu reálneho rozmeru (napr. n = 80, m = 20) je to ale už bežnými nástrojmi na riešenie celočíselných úloh nezvládnuteľný rozmer.

2. ***Heuristický prístup*** naprogramovaný klasickým procedurálnym prístupom (napr. v Pascale, alebo v jazyku C).

   Pre každý možný výber distribučných centier (tých je 2^m) je priradenie odberateľov triviálne (každého odberateľa preradíme najbližšiemu distribučnému centru). Vývoj programu v Pascale trval pre túto aplikáciu v tomto prípade cca. 2 mesiace. Pre pomerne malú zmenu zadania (napr. že odberateľ môže odoberať aj z viacerých distribučných centier naraz) je nutné program úplne zmeniť.

3. Pri použití ***logického programovania ohraničení***, ako deklaratívneho programovacieho prístupu, trvalo naprogramovanie tej istej úlohy trvalo cca. 2 týždne a výsledný program bol omnoho flexibilnejší. Takže napr. zmena zadania si vyžiada iba jednoduchú zmenu programu. Výborný prototypovací nástroj.

   Program pre riešenie tejto úlohy v jazyku ECLiPSe možno nájsť v knihe [Mach, Paralič 2000].

Na tomto mieste uvedieme iný typ deklaratívneho programu na riešenie tejto úlohy, v prostredí VisualXpress, čo je nástroj na riešenie úloh lineárneho a celočíselného programovania. Uvádzame tri verzie tohto programu, každú pre trocha modifikovanú verziu zadania úlohy.

Ako prvá je uvedená základná verzia úlohy tak, ako bola popísaná vyššie. Distribučné centrá (DC) bez obmedzenia kapacity distribučného centra, každému odberateľovi (O) dodáva práve jedno distribučné centrum.

```
LET
d=3   !miesta pre distribučné centra
o=5   !odberatelia

TABLES
vzdialenosti(d,o)
fixne_naklady(d)
dodavky(o)

DATA
vzdialenosti(1,1) = 5,3,8,4,2
vzdialenosti(2,1) = 9,6,1,3,5
vzdialenosti(3,1) = 2,4,6,8,3

fixne_naklady(1) = 300, 200, 400
dodavky(1) = 50, 70, 30, 80, 60

VARIABLES
y(d)   !zriadiť, alebo nezriadiť distribučné centrum
x(d,o) !bude dané DC dodávať danému odberateľovi (áno/nie)
```

```
CONSTRAINTS
odberatelia(j=1:o): SUM(i=1:d) x(i,j) = 1
dodavatelia(i=1:d, j=1:o): x(i,j) < y(i)
naklady: SUM(i=1:d) fixne_naklady(i)*y(i) +
         SUM(i=1:d, j=1:o) dodavky(j)* vzdialenosti(i,j)*x(i,j)$

BOUNDS
y(i=1:d) .BV.
x(i=1:d, j=1:o) .BV.
```

Druhá verzia programu uvažuje už distribučné centrá s obmedzenou kapacitou, ale každému odberateľovi stále dodáva všetok tovar práve jedno distribučné centrum.

```
LET
d=3     !miesta pre distribučné centra
o=5     !odberatelia

TABLES
vzdialenosti(d,o)
fixne_naklady(d)
kapacity(d)
dodavky(o)

DATA
vzdialenosti(1,1) = 5,3,8,4,2
vzdialenosti(2,1) = 9,6,1,3,5
vzdialenosti(3,1) = 2,4,6,8,3

fixne_naklady(1) = 300, 200, 400
kapacity(1) = 150, 130, 170
dodavky(1) = 50, 70, 30, 80, 60

VARIABLES
y(d)   !zriadiť, alebo nezriadiť distribučné centrum
x(d,o) !bude dané DC dodávať danému odberateľovi (áno/nie)

CONSTRAINTS
odberatelia(j=1:o): SUM(i=1:d) x(i,j) = 1
dodavatelia(i=1:d, j=1:o): x(i,j) < y(i)
kapacita(i=1:d): SUM(j=1:o) x(i,j)* dodavky(j) < kapacity(i)
naklady: SUM(i=1:d) fixne_naklady(i)*y(i) +
         SUM(i=1:d, j=1:o) dodavky(j)* vzdialenosti(i,j)*x(i,j)$

BOUNDS
y(i=1:d) .BV.
x(i=1:d, j=1:o) .BV.
```

Posledná verzia programu je najvšeobecnejšia. Distribučné centrá sú s obmedzenou kapacitou a jeden odberateľ môže odoberať tovar z viacerých distribučných centier naraz. V tomto programe už premenné x_{ij} nie sú binárne, ale obsahujú množstvo tovaru dodávaného i-tym distribučným centrom j-temu odberateľovi.

```
LET
d=3     !miesta pre distribučné centra
o=5     !odberatelia

TABLES
vzdialenosti(d,o)
fixne_naklady(d)
kapacity(d)
dodavky(o)

DATA
vzdialenosti(1,1) = 5,3,8,4,2
vzdialenosti(2,1) = 9,6,1,3,5
vzdialenosti(3,1) = 2,4,6,8,3

fixne_naklady(1) = 300, 200, 400
kapacity(1) = 150, 130, 170
dodavky(1) = 50, 70, 30, 80, 60

VARIABLES
y(d)  !zriadiť, alebo nezriadiť distribučné centrum (binárne)
x(d,o)!koľko bude dané DC dodávať danému odberateľovi (reálne čísla)

CONSTRAINTS
odberatelia(j=1:o): SUM(i=1:d) x(i,j) = dodavky(j)
kapacita(i=1:d): SUM(j=1:o) x(i,j) < kapacity(i)*y(i)
naklady: SUM(i=1:d) fixne_naklady(i)*y(i) +
         SUM(i=1:d, j=1:o) vzdialenosti(i,j)*x(i,j)$

BOUNDS
y(i=1:d) .BV.
```

4 Prognózovanie

4.1 Základné pojmy

Plánovanie je nevyhnutnou súčasťou práce manažéra, prognózovanie mu z časti pomáha redukovať neurčitosť pri vytvorení konkrétnych plánov [Malindžák 1996], t.j. odhad budúcich objednávok.

Cieľom prognózovania je najčastejšie odhad predaja výrobkov, tj. budúcich požiadaviek na výrobu, od čoho sa následne odvíja odhad spotreby materiálov, energie a ďalších zdrojov. Ale prognózovať možno aj vývoj cien, inflácie a pod.

```
Postup pri prognózovaní:

1. Určenie cieľa prognózy.

2. Určenie časového horizontu prognózy.

3. Výber metódy prognózovania.

4. Zber a analýza vhodných informácií a ich spracovanie pre
   účely prognózy.

5. Monitoring prognózy.
```

Metódy prognózovania delíme na dve hlavné skupiny:

a) Kvantitatívne metódy

- sú založené na analýze a spracovaní historických údajov a ich extrapolácii na obdobie prognózy, alebo
- na hľadaní kauzálnych vzťahov časovej rady údajov použitých pre prognózovanie.

b) Kvalitatívne metódy

- sú založené viac na subjektívnych informáciách zákazníkov, predajcov, manažérov, expertov, na základe ktorých sa následne robí numerický odhad.
- Používajú sa vtedy, ak je prognóza potrebná rýchlo, alebo ak nemáme dostatok informácií pre použitie kvantitatívnych metód.

4.2 *Kvantitatívne metódy prognózovania*

Tieto metódy využívajú už spomínané časové rady údajov. Časová rada údajov je časovo usporiadaná sekvencia údajov, pozorovaní, získaných v nejakom pravidelnom (rovnakom) časovom intervale (napr. deň, mesiac, rok a pod.).

Princíp je v tom, že sa predpokladá nasledovné: vývoj a vzťahy medzi hodnotami v minulosti budú pokračovať aj v budúcnosti.

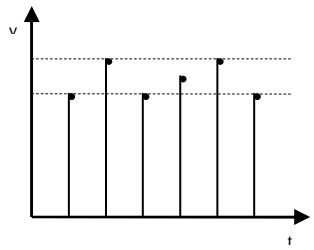
Dátovú množinu je možné v niektorých prípadoch rozšíriť o ďalšie vstupujúce atribúty a budovať *prediktívny model* (viď. napr. [Paralič 2003]).

Dôležitými faktormi prognóz časového radu sú napr. ***časový horizont prognózy***, t.j. na koľko intervalov dopredu je potrebné stanoviť prognózu. O strategickej prognóze hovoríme pri 3 – 5 časových intervaloch (kríza alebo konjunktúra skupiny výrobkov, požiadavky na nové výrobky, výkony nadväzujúcich odvetví, vývoj cien vstupov …) O taktickej prognóze zase ak ide o jeden časový interval dopredu (pre účely kapacitného plánovania, objednávok vstupov s dlhými dodacími lehotami, pre účely operatívneho plánovania …)

Ďalším takýmto faktorom je otázka, **koľko hodnôt prognózovanej veličiny do minulosti vziať do úvahy** – rozsah hodnôt a ich závažnosť. Z hľadiska štatistiky by ich malo byť čím viac, ale na druhej strane čím staršie údaje použijeme, tým menší je ich vplyv na prognózovanú hodnotu. Preto je potrebné nájsť správnu mieru medzi protichodnými požiadavkami štatistiky a aktuálnosti dát.

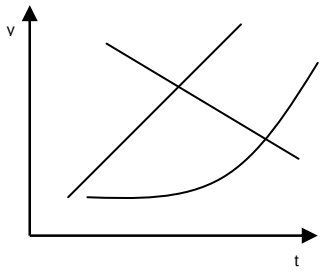
Z hľadiska chovania sa časovej rady údajov poznáme štyri základné typy (modely) chovania sa časovej rady údajov.

1. ***Konštantný model*** (K-model) je charakteristický tým, že v dlhom časovom období sa sledované hodnoty pohybujú iba v úzkom intervale.



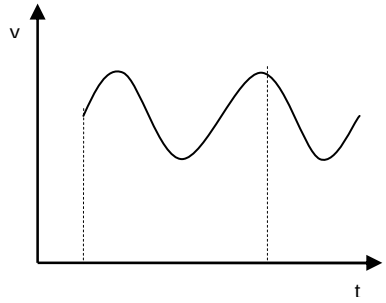
Obr. 11 Konštatný model chovania sa časovej rady údajov

2. ***Trendový model*** (T-model) – krivka má jasný smer – vyznačuje sa trvalou kvalitou zmeny.



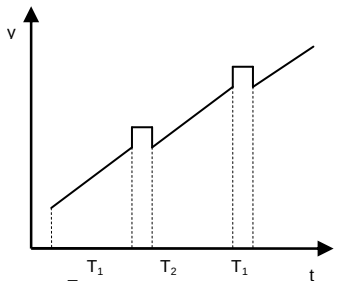
Obr. 12 Príklady trendových modelov chovania sa časovej rady údajov

3. ***Cyklický model*** (C-model) je charakteristický tým, že po určitom čase sa hodnoty zhruba opakujú.



Obr. 13 Príklad cyklického modelu chovania sa časovej rady údajov

4. ***Sezónny model*** (S-model) je podobný cyklickému, ale podobnosť (opakovanie hodnôt) súvisí s obdobím, period teda môže byť viac a rôznej dĺžky.



Obr. 14 Príklad sezónneho trendového modelu

5. ***Kombinovaný model*** – je kombináciou druhého typu modelu s tretím alebo štvrtým (cyklický trendový, alebo sezónny trendový model).

Spomenieme niekoľko jednoduchých metód prognózovania. Prvé tri sú založené na výpočte priemernej hodnoty. Y_i označujeme hodnotu prognózovanej veličiny z časovej rady údajov za interval i.

A) Aritmetický priemer vypočítame nasledovne:

$$Y_{n+1} = \frac{\sum_{i=1}^{n} Y_i}{n}$$

B) Kĺzavý priemer – do prognózy sa zahŕňa len m posledných hodnôt, nie staršie hodnoty, t.j.:

$$Y_{n+1} = \frac{1}{n-m} \sum_{i=1+m}^{n} Y_i$$

C) Vážený priemer zohľadňuje aktuálnosť údajov časovej rady pridelením váh. Čím staršie údaje, tým nižšia váha. Vypočíta sa nasledovne:

$$Y_{n+1} = \frac{\sum_{i=1}^{n} Y_i w_i}{\sum_{i=1}^{n} w_i}$$

D) Exponenciálne vyrovnávanie – zohľadňuje vplyv chyby prognózy z predchádzajúcej prognózy.

$Y_n = Y_{n-1} + \alpha(A_{n-1} - Y_{n-1}) = \alpha A_{n-1} + Y_{n-1}(1-\alpha)$, kde Y_{n-1} je prognózou v predchádzajúcom intervale a A_{n-1} je skutočná hodnota v tomto intervale. Táto metóda je citlivá na voľbu hodnoty α. Čím väčšia hodnota α, tým väčší vplyv na prognózu má skutočná hodnota prognózovanej veličiny z predchádzajúceho intervalu.

E) Lineárna regresia vypočítava parametre priamky, ktorá najlepšie aproximuje množinu bodov (minimalizuje súčet štvorcov odchýliek daných bodov od regresnej priamky). Ak sú dané trénovacie dáta vo forme bodov (x_1, y_1), (x_2, y_2), ..., (x_n, y_n), potom regresné koeficienty α, β možno odhadnúť uvedenou metódou pomocou nasledovných vzťahov:

$\beta = \frac{\sum_{i=1}^{n}(x_i - \bar{x})\cdot(y - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$, $\alpha = \bar{y} - \beta \cdot \bar{x}$, kde $\bar{x}$ je priemer hodnôt $x_1, x_2, ..., x_n$ a $\bar{y}$ je priemer hodnôt $y_1, y_2, ..., y_n$.

F) Metóda harmonických váh vychádza z poslednej (najčerstvejšej) hodnoty v časovej rade údajov, ktorú násobí pomerovým indexom. Do tohto pomerového indexu sú premietnuté staršie hodnoty z časovej rady údajov nasledovným spôsobom:

$PI = \frac{Y(t)}{Y(t-1)}$ pomerový index

$\overline{PI} = \sqrt[n-1]{\prod_{t=2}^{n} PI(t)^{w(t)}}$ výsledný pomerový index

$w_t = \sum_{j=1}^{t-1} \frac{1}{n-j}$ harmonické váhy

$Y_{n+1} = \overline{PI}.Y_n$

Príklad

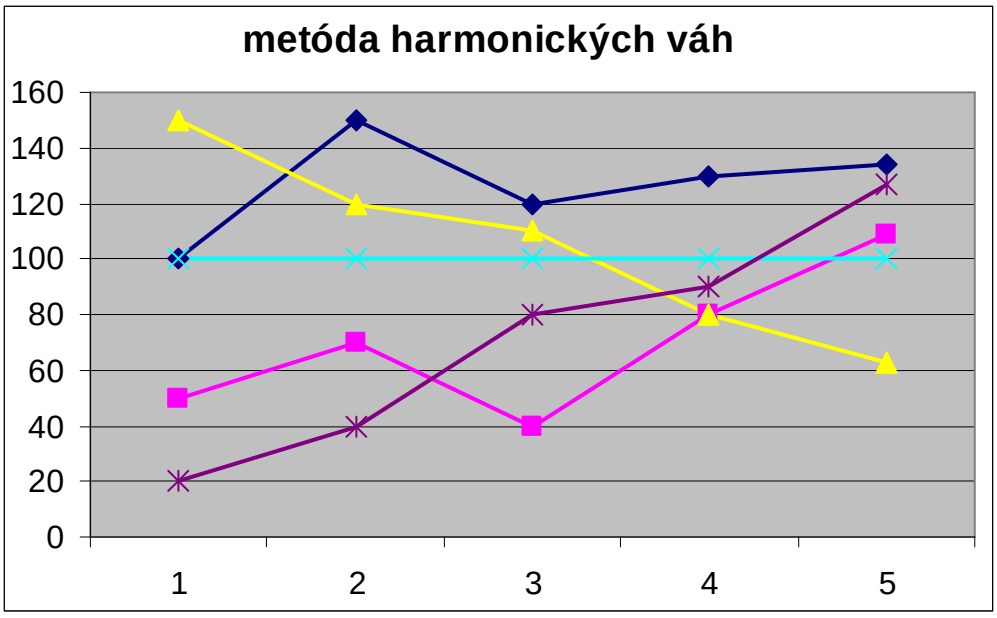
Firma vyrába 5 druhov výrobkov. Porovnajte hodnoty za roky (n - 3) až n a vypočítajte prognózu na rok n +1. Vyberte vhodnú metódu. Zadané hodnoty údajov časovej rady pre každý druh výrobkov, ako aj prognózy vypočítané na základe týchto údajov vyššie uvedenými metódami sú uvedené v Tab. 1. Všetky hodnoty prognózy boli vypočítané za pomoci programu MS Excel, v ktorom je potrebné okrem výpočtov zostrojiť aj grafy. Tieto umožnia zhodnotiť charakter priebehu časovej rady údajov a vybrať najvhodnejšiu

metódu, ktorá sa pre daný priebeh najlepšie približuje danému typu priebehu (napr. konštantný, trendový, sezónny a pod.). V Tab. 1 sú hrubo zvýraznené podľa nášho uváženia najlepšie prognózy, ktoré boli vybrané na základe zostavených grafov.

výrobky	n-3	n-2	n-1	n	AP	KP	VP	EV	LR	MHV
1	100	150	120	130	125	**133**	128	128	143	134
2	50	70	40	80	**60**	63	63	72	78	109
3	150	120	110	80	115	103	104	86	**50**	63
4	100	100	100	100	100	100	100	100	100	100
5	20	40	80	90	58	70	70	88	131	**127**

Tab. 1 Zadané časové rady údajov pre jednotlivé typy výrobkov a prognózy vypočítané podľa metód uvedených vyššie

Pre ilustráciu vyberáme graf so zobazením vývoja časových rád údajov za všetky typy výrobkov a prognózou na ďalšie obdobie (posledný vyobrazený bod, v čase 5) vypočítanou metódou harmonických váh (viď. Obr. 15).



Obr. 15 Časové rady údajov a prognózy metódou harmonických váh pre 5 druhov výrobkov zadaných v ilustračnom príklade

Z uvedených grafov je možné si všimnúť, že metóda harmonických váh je pomerne úspešná pri trendových modeloch. Pre konštatntné modely je úplne postačujúca niektorá z metód založených na priemerovaní (aritmetický, kĺzavý, alebo vážený priemer). Pri trendových lineárnych modeloch je výborná lineárna regresia. Exponenciálne vyrovnávanie dokáže tiež zachytiť určitý trend, ale je silne závislé na kvalite poslednej prognózy a výbere koeficientu α.

4.3 Kvalitatívne metódy prognózovania

V tejto časti uvedieme stručný popis štyroch rôznych kvalitatívnych metód, ich výhody a nevýhody.

4.3.1 Odhad predajcov

Prognóza sa uskutoční priamo na základe odhadov predajcov (čo a koľko budú zákazníci v najbližšom období kupovať).

Výhody:

- predajcovia sú jednými z najkompetentnejších (informácie z prvej ruky)
- je známe ich geografické rozloženie

Nevýhody:

- ľudský faktor, t.j. individualita predajcov
- subjektívne posúdenie toho, či sa zákazník na tovar iba informuje, alebo má reálny záujem tovar aj kúpiť
- snaha o vykreslenie reality v lepšom svetle

4.3.2 Skupinový posudok

Zakladá sa na vedomostiach a skúsenostiach odborníkov pracujúcich v danej oblasti (manažéri, obchodníci, technickí pracovníci). Títo sa stretnú a konsenzuálne vypracujú prognózu. Využíva sa komparatívny prístup a analógia. Vhodné napr. pri zavádzaní nových výrobkov.

Výhody:

- rýchlosť
- zastúpenie všetkých skupín ľudí, ktorí do toho majú čo povedať

Nevýhody:

- závisí na schopnosti komunikovať, počúvať, dohodnúť sa

4.3.3 Prieskum trhu

Je to systematický prístup vytvárania a testovania hypotéz o trhu. Používa sa najmä pri zavádzaní nových výrobkov.

Postup pri prieskume trhu:

1. Návrh dotazníka – vždy sú tam dve skupiny otázok (ekonomické a demografické údaje o respondentovi a otázky súvisiace s jeho záujmom o nový výrobok).
2. Výber spôsobu komunikácie – osobne, telefonicky, e-mail, klasická pošta
3. Výber reprezentatívnej vzorky respondentov – náhodný výber z potenciálnej skupiny zákazníkov
4. Realizácia prieskumu a spracovanie zozbieraných údajov
5. Vytvorenie prognózy za základe zozbieraných a zosumarizovaných údajov

---

Výhody:

- exaktnosť (dá sa získať množstvo informácií)
- informácie z prvej ruky

Nevýhody:

- vyššie náklady
- dlhšie trvanie

4.3.4 Metóda DELPHI

Zakladá sa na procese dosiahnutia dohody medzi koordinačnou skupinou odborníkov (*Delphi committee - DC*) a medzi anonymnou skupinou expertov (ktorí medzi sebou o riešení prognózy nekomunikujú). Používa sa najmä na dlhodobé prognózy, najmä nových výrobkov.

```
Postup pri metóde DELPHI:
1. DC určí spôsob komunikácie a formuluje otázky na ktoré žiada
   odpoveď od expertov.
2. Experti sformulujú svoje odpovede aj so zdôvodnením a pošlú DC.
3. DC spracuje stanoviská expertov a vytvorí spoločné stanovisko
   (konsenzus) – 1. variant prognózy a rozošle ho expertom.
4. Proces sa opakuje až kým sa nedospeje ku konsenzu všetkých
   expertov.
```

Výhody:

- Konsenzus skupiny nezávislých expertov

Nevýhody:

- Môže trvať dlho
- Nemusí byť zaručená anonymita expertov
- Zle formulované otázky môžu viesť k zlým záverom

4.4 Chyby prognózy

Chyba prognózy (E_t) je rozdiel medzi skutočnými požiadavkami na výrobu za obdobie t (A_t) a prognózou požiadaviek na toto obdobie (Y_t), t.j.

$E_t = A_t - Y_t$

Existujú rôzne kritériá hodnotenia presnosti prognózy z dlhodobého hľadiska, napr. za účelom porovnania rôznych metód prognózovania, resp. pre monitorovanie prognózy.

1) Kumulatívna chyba prognózovania

$$CFE = \sum_{t=1}^{n} E_t$$

2) Kvadratická chyba prognózovania - najčastejšia

$$MSE = \frac{\sum_{t=1}^{n} E_t^2}{n}$$

3) Štandardná odchýlka chyby prognózovania

$$\sigma = \frac{\sqrt{\sum_{t=1}^{n} E_t^2}}{n}$$

4) Priemerná absolútna chyba prognózovania

$$MAD = \frac{\sum_{t=1}^{n} |E_t|}{n}$$

5) Priemerná absolútna percentuálna chyba prognózovania

$$MAPE = \frac{\frac{\sum_{t=1}^{n} |E_t|}{A_t}}{n}$$

5 Plánovanie výrobných kapacít

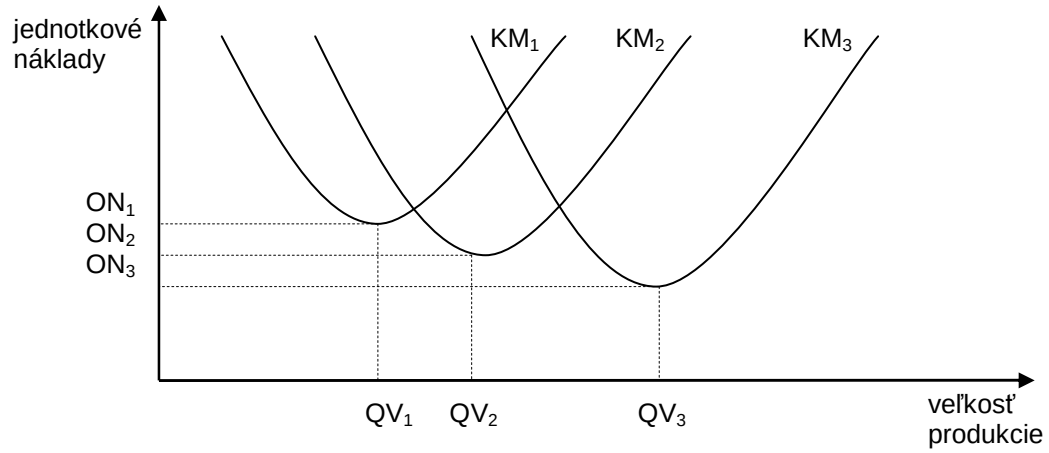
Manažment výrobných kapacít má za cieľ dosiahnuť maximálny súlad (rovnováhu) medzi kapacitami výrobného procesu a kapacitnými nárokmi vyplývajúcimi z požiadaviek naň.

K manažmentu výrobných kapacít patria najmä tieto tri úlohy:

1. Určenie veľkosti výrobnej kapacity
2. Stanovenie kapacitnej stratégie
3. Kapacitné vyváženie výrobného procesu

5.1 Určenie veľkosti výrobnej kapacity

Veľkosť výrobnej kapacity závisí jednak od veľkosti firmy a tiež od veľkosti produkcie. Závislosť jednotkových výrobných nákladov od týchto faktorov znázorňuje nasledujúci Obr. 16.



Obr. 16 Závisloť jednotkových výrobných nákladov od veľkosti produkcie a veľkosti firmy

Z uvedených grafov vyplýva, že väčšie firmy môžu dosiahnuť nižšie jednotkové výrobné náklady, ale potrebujú pritom väčší objem výroby.

Kapacita stroja je disponibilný čas, ktorý má daný stroj k dispozícií na výrobu určitých výrobkov.

Výrobná kapacita je maximálna produkcia za časovú jednotku.

Úzke miesto výrobného procesu je tá výrobná operácia (stroj, prevádzka), na ktorej sa kapacita vyčerpá ako prvá pri zvyšovaní produkcie.

Maximálna kapacita je limitovaná úzkym miestom, to znamená že **výrobná kapacita sa rovná kapacite úzkeho miesta**, ktoré je možné vypočítať napr. pomocou statického výpočtu.

Označme n počet výrobkov, ktoré sa v danej prevádzke vyrábajú, pričom pre každý i-ty typ výrobku je potrebné vyrobiť množstvo M_i $(i = 1, 2, \ldots, n)$. V prevádzke sa nachádza m strojov (resp. častí výroby), na ktorých prebieha výroba daných výrobkov. Čas potrebný na spracovanie jednotkového

množstva i-teho výrobku na j-tom stroji označme t_{ij}. KN_j budú kapacitné nároky na j-ty stroj a KM_j zase kapacitné možnosti j-teho stroja. Kapacitné nároky vypočítame nasledovne:

$$KN_j = \sum_{i=1}^{n} M_i t_{ij}$$

Kapacitné možnosti daného stroja sú dané jeho výrobcom. Ak sú kapacitné možnosti jednotlivých strojov rovnaké, tj. KM_j je rovnaké pre $\forall\ j = 1, 2, \ldots, m$, potom úzke miesto výrobného procesu bude:

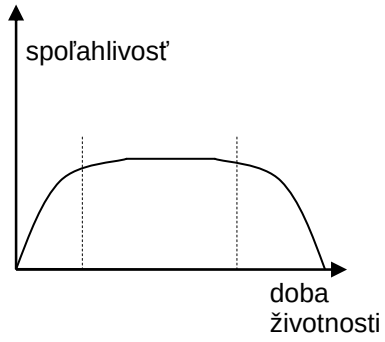
$$J_{ú} = \max_j \{KN_j\}$$

Ak sú kapacitné možnosti jednotlivých strojov (častí prevádzky) rôzne, potom úzke miesto určíme nasledovne:

$$J_{ú} = \min_j \{KM_j - KN_j\}$$

Pri stanovení kapacitných možností KM_j je potrebné brať ohľad aj na dve ďalšie dôležité skutočnosti, a síce že iba 75% – 90% skutočnej maximálnej kapacity stroja možno vyčleniť pre KM_j, zvyšok slúži na regulačné účely.

Pri kalkulácii KM_j treba zohľadniť aj spoľahlivosť strojov vyplývajúcu z doby ich prevádzky, vzhľadom na ich životnosť. Jej priebeh vyjadruje nasledujúci graf (viď. Obr. 17).



Obr. 17 Závislosť spoľahlivosti stroja (kalkulovateľnej časti KM_j) od jeho životnosti

Príklad

Majme dva stroje (dve časti výrobného procesu – S_1, S_2) a tri druhy výrobkov (V_1, V_2, V_3), ktoré sa v danom výrobnom procese vyrábajú. Časy spracovania jedného kusu každého typu výrobku na každom stroji sú dané v tabuľke Tab. 2 (v minútach).

Ktorý zo strojov (častí výrobného procesu) je úzkym miestom v prípade výroby 100 ks V_1, 150 ks V_2 a 130 ks V_3, ak:

a) kapacitné možnosti oboch strojov sú rovnaké
b) kapacitné možnosti sú rôzne (S_1 má kapacitné možnosti 3000 minút a S_2 má kapacitné možnosti 2800 minút)?

t_{ij}	S_1	S_2
V_1	5	8
V_2	7	6
V_3	9	7

Tab. 2 Zadanie časov spracovania výrobkov na jednotlivých strojoch

Riešenie

Počet strojov j = 2, počet výrobkov i = 3, požiadavky na výrobu v počtoch výrobkov jednotlivých druhov sú: M_1 = 100, M_2 = 150, M_3 = 130. Jednotkové časy spracovanie výrobku i na stroji j (t.j. t_{ij}) sú uvedené v tabuľke Tab. 2.

Kapacitné nároky na prvý stroj (KN_1) budú:

$$KN_1 = \sum_{i=1}^{3} M_i \cdot t_{i1} = 100 \cdot 5 + 150 \cdot 7 + 130 \cdot 9 = 2720$$

Kapacitné nároky na druhý stroj (KN_2) budú:

$$KN_2 = \sum_{i=1}^{3} M_i \cdot t_{i2} = 100 \cdot 8 + 150 \cdot 6 + 130 \cdot 7 = 2610$$

a) $KM_1 = KM_2$, takže $J_{ú} = \max_j \{KN_1, KN_2\}$ = max{2720, 2610} = **1 (t.j. prvý stroj)**

b) KM_1 = 3000, KM_2 = 2800, takže $J_{ú} = \min_j \{KM_1 - KN_1, KM_2 - KN_2\}$ =

min{3000 - 2720, 2800 - 2610} = min{280, 190} = **2 (t.j. druhý stroj)**

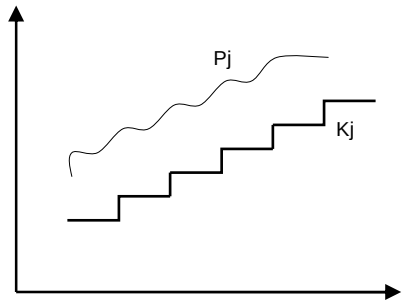
5.2 Stanovenie kapacitnej stratégie

Kapacitná stratégia hovorí o tom, akým spôsobom podnik pristupuje z dlhodobého hľadiska k pomeru medzi kapacitou výroby K_j a skutočnými požiadavkami (resp. prognózou) P_j.

$KV_j = K_j - P_j$ je ***hlavný kapacitný vzťah***. Pritom sú nasledovné možnosti:

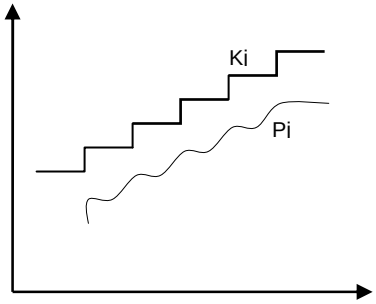
- Ak sa firma snaží udržiavať $KV_j > 0$, t.j. ak kapacita výroby prevyšuje požiadavky (to zodpovedá agresívnej stratégii)
- $KV_j = 0$ ak je kapacita rovná požiadavkám (je ťažké dosiahnuť presnú rovnosť)
- Ak sa firma snaží udržiavať $KV_j < 0$, t.j. kapacita výroby je nižšia ako požiadavky na ňu (to zodpovedá konzervatívnej stratégii)

Konzervatívna kapacitná stratégia (angl. „wait and see“) je znázornená na Obr. 18.



Obr. 18 Konzervatívna stratégia, kapacity sú zvyšované s oneskorením po náraste požiadaviek na výrobu

Agresívna kapacitná stratégia (angl. „preempt of competition“) je znázornená na Obr. 19.



Obr. 19 Agresívna stratégia, navyšovanie kapacít predbieha skutočný nárast požiadaviek na výrobu

5.3 Kapacitné vyváženie výrobného procesu

Cieľom je dosiahnuť požadovanú hodnotu kapacitného vzťahu v zmysle podnikovej kapacitnej stratégie. Podnik môže pritom rôznymi spôsobmi ovplyvňovať obe veličiny, tj. K_j aj P_j.

Prispôsobenie výrobnej kapacity K_j sa týka rôznych typov kapacít, ktoré je možné rozdeliť na jednorázové (suroviny, resp. materiál) a trvalé (ľudia, stroje). Pre podnik je ťažšie ovplyvňovať požiadavky na výrobu P_j ale aj to je do istej miery možné. Niektoré z možností ako ovplyvňovať úroveň kapacít, resp. požiadaviek priebežne, sú zhrnuté v Tab. 3.

Typ Kj	Zvýšenie Kj	Zníženie Kj
Materiál	- minimálna spotreba materiálu na jednotku výroby - náhrada inými materiálmi - zvýšenie plánovaných zásob - presun z iných výrobných procesov	- zníženie plánu zásob - presun na iné výrobné procesy
Stroje	- skrátenie výrobných časov - minimalizovanie priestorov (skrátenie udržby) - automatizácia výroby - nové stroje (kapacity) - sprostredkovanie výrobkov od iného výrobcu	- štandardná výroba - zvýšenie plánu udržby - zaradenie na iné práce
Ľudské zdroje	- nadčasy (2., 3. smena) - zmena plánov dovoleniek - presun ľudí z iných výrobných procesov - prijatie nových zamestnancov	- preplánovanie dovoleniek - čiastočne obmedzenie pracovnej doby - zníženie počtu prác - preradenie na inú prácu
Požiadavky na výrobu	- zvýšenie intenzity reklamy - presun niektorých zákaziek z nasledujúceho obdobia do súčasnosti	- odmietnutie reálnych zákaziek - rozdelenie veľkých zákaziek do budúcich období - presun zákaziek do ďalšieho obdobia

Tab. 3 Príklady opatrení ovplyvňujúcich aktuálne kapacity rôznych typov, ako aj opatrenia na ovplyvnenie požiadaviek na výrobu

6 Rozvrhovanie

6.1 Základné charakteristiky úloh rozvrhovania

Každá úloha rozvrhovania musí mať definované základné parametre tzv. povinných charakteristík, ku ktorým patrí charakteristika strojov (procesorov) a úloh, ktoré je potrebné na týchto strojoch (procesoroch) rozvrhnúť. Okrem týchto základných charakteristík často pristupujú aj niektoré ďalšie charakteristiky, najmä rôzne typy ohraničení, prípadne definovanie zdrojov, ktoré sa úlohami spotrebúvajú.

Takže zhrnieme charakteristiky úloh rozvrhovania, ktoré budú ďalej podrobnejšie vysvetlené.

I. **Hlavné charakteristiky**
   1. Stroje (procesory)
   2. Úlohy

II. **Doplnkové charakteristiky**
   A) precedenčné ohraničenia (usporiadanie)
   B) disjunktné ohraničenia (zdieľanie strojov/procesorov)
   C) zakázky (*jobs*)
   D) pomocné zdroje (*resources*)

Najskôr sa pozrime bližšie na ***charakteristiku strojov, resp. procesorov***. Poznáme dva základné typy strojov (procesorov) z hľadiska rozvrhovania:

a) *paralelné* - úloha môže bežať na ľubovoľnom stroji
b) *dedikované (špecializované)* - úloha môže bežať len na špeciálne určenom stroji

Podľa výkonnosti procesorov ich možno rozdeliť na:

a) *identické* – na každom stroji trvá vykonanie danej úlohy rovnako dlho
b) *uniformné* – každý stroj (procesor) má svoju rýchlosť, ktorá nezávisí na úlohe, takže vnáša při spracovaní úloh konštantné zrýchlenie (resp. spomalenie), ktoré označíme b_j
c) *nesúvzťažné* – rýchlosť stroja (procesora) závisí na vykonávanej úlohe, tj. čas vykonania úlohy i na stroji j bude t_{ij}

Čo sa týka druhej hlavnej charakteristiky, t.j. ***úloh***, predpokladáme že je zadaná množina úloh $T = \{T_1, T_2, \ldots, T_n\}$. Povinné zadané údaje o jednotlivých úlohách sú:

1) *Čas spracovania úlohy* t_i (dĺžka trvania – *task duration*) – vo všeobecnosti to môže byť vektor (pre každý stroj/procesor iná dĺžka trvania) $[t_{i1}, t_{i2}, \ldots, t_{im}]$
   - v prípade *identických* strojov ide vlastne iba o jednu hodnotu t_i, t.j. $t_{ij} = t_i$ pre všetky stroje/procesory $j = 1 .. m$
   - v prípade uniformných procesorov $t_{ij} = t_i / b_j$
   - pre nesúvzťažné procesory je rôzne t_{ij}

2) *Čas pripravenosti* r_i (*release time*) – od akého okamžiku je úloha pripravená na realizáciu. Ak sú všetky úlohy pripravené naraz, potom $r_i = 0$ (pre všetky $i = 1 .. n$)
3) *Požadovaná doba splnenia* d_i (*due date*) - doba dokedy by mala byť úloha splnená.

Nepovinné údaje, ktoré o úlohách ešte tiež môžu byť zadané:

4) *Dodacie časy* τ_i (*deadline*) – neprekročiteľné časy určenia.
5) *Priority* w_i – významnosť úlohy

Ohraničenia možno rozdeliť na dve hlavné skupiny. ***Precedenčné ohraničenia*** definujú usporiadania medzi úlohami (spravidla dvojicami úloh). Pre úplný zoznam 16 typov precedenčných ohraničení podľa Alena pozri napr. [Paralič 1997].

Veľmi vhodnou pomôckou pre grafické znázornenie precedenčných ohraničení je *precedenčný graf*. Úlohy sú reprezentované uzlami v grafe. Precedencie sú v ňom reprezentované orientovanými hranami.

Príklad

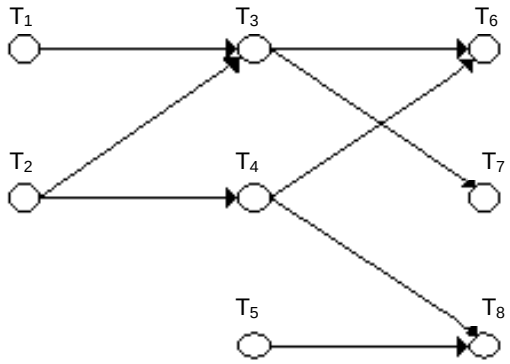
Majme 3 identické paralelné procesory a daných 8 úloh s týmito parametrami:

- $m = 3$, paralelné procesory, $P = \{P1, P2, P3\}$
- $n = 8$, úlohy, $T = \{T1, T2, \ldots, T8\}$
- $t_i = [3, 4, 1, 2, 1, 2, 3, 2]$ – časy spracovania úloh
- $r_i = 0$ (i = 1, ... , 8) – časy pripravenosti
- $d_i = [5, 4, 5, 3, 7, 6, 9, 12]$ – požadované časy ukončenia úloh
- $w_i = [1, 2, 1, 3, 1, 2, 2, 2]$ – priority úloh
- $\{T1 < T3, T2 < T3, T2 < T4, T3 < T6, T3 < T7, T4 < T6, T4 < T8, T5 < T8\}$ – precedencie

Riešenie

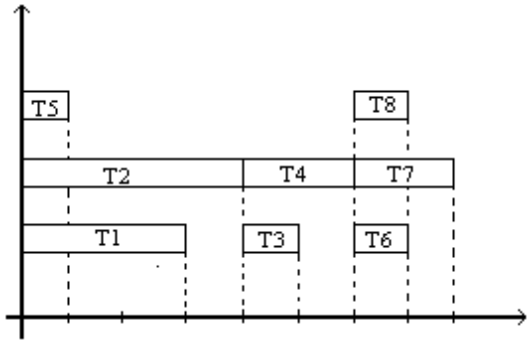
Zostrojme najprv precedenčný graf. Graf konštruujeme postupne, pričom si uzly rozdelíme na vstupné, výstupné a medziľahlé. Vstupné sú tie, ktoré sa v precedenciách nachádzajú iba na ľavej strane. Výstupné sa v nich nachádzajú iba na pravej strane. Medziľahlé sa vyskytujú na pravej i na ľavej strane precedencií. Výsledný graf je zobrazený na Obr. 20.

Teraz môžme začať konštruovať rozvrh manuálne, vo forme Ganttovho diagramu. ***Ganttov diagram*** je grafickou reprezentáciou rozvrhu (os x reprezentuje čas, na osi y sú jednotlivé stroje/procesory). Rozvrh môžeme skonštruovať na základe precedenčného grafu, pričom dodržiavame precedencie a úlohy zaradzujeme na voľný stroj v najskoršom možnom čase.



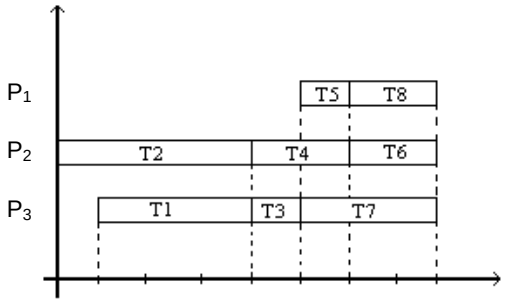
Obr. 20 Precedenčný graf zadanej úlohy

Výsledkom takéhoto postupu je rozvrh reprezentovaný Ganttovým diagramom na Obr. 21.



Obr. 21 Ganttov diagram rozvrhu zostrojeného podľa precedenčného grafu (rozvrh *R1*)

Iný rozvrh zostrojený tým istým postupom, ale so snahou minimalizovať prestoje strojov je zobrazený na Obr. 22.



Obr. 22 Ganttov diagram rozvrhu zostrojeného podľa precedenčného grafu (rozvrh *R2*)

Druhým typom ohraničení sú ***disjunktné ohraničenia***, ktoré vyplývajú zo zdieľania strojov (procesorov) úlohami. Na jednom stroji totiž môže byť vykonávaná naraz iba jedna úloha, takže ak na danom procesore musí byť

spracovaných viacero úloh, vznikajú medzi nimi precedencie vo forme disjunkcií (viď. príklad uvedený nižšie). Práve tieto ohraničenia sú zdrojom kombinatorickej explózie pri riešení úloh rozvrhovania.

Rozvrh* (*R*)** je súbor údajov, z ktorého je zrejmé, v ktorých časových intervaloch sa majú jednotlivé úlohy realizovať. Nech $c_i(R)$ je čas ukončenia úlohy T_i v rozvrhu R. Potom je zrejmé, že každý ***prípustný rozvrh je daný n-ticou $[c_1(R), \ldots, c_n(R)]$, pričom platí, že spĺňa všetky ohraničenia.

Kriteriálna funkcia je definovaná na množine všetkých prípustných rozvrhov spravidla ako nejaká reálna funkcia f času ukončenia jednotkových úloh, t.j. $F(R) = f(C_1(R), \ldots, C_n(R))$.

Optimálny rozvrh je taký prípustný rozvrh, pre ktorý daná kriteriálna funkcia f nadobúda minimum na množine všetkých prípustných rozvrhov.

Kriteriálna funkcia je regulárna vtedy, ak nie je možný jej nárast bez toho, aby sa nepredĺžil termín ukončenia aspoň jednej úlohy.

Dominantná množina rozvrhov (Dom) je taká množina, že pre každý rozvrh R, ktorý nie je z dominantnej množiny Dom existuje taký rozvrh S z dominantnej množiny Dom, že pre každú úlohu i $(i = 1 .. n)$ platí $c_i(R) \le c_i(S)$ (tj. že v rozvrhu S nezačína neskôr ako v rozvrhu R). Formálne sa to dá zapíať takto: $\forall R \notin Dom : \{\exists S \in Dom,\ \ c_i(S) \le c_i(R),\ \ \forall i = 1,\ \ldots\ , n\}$.

Najčastejšie používané kriteriálne funkcie (pričom f_i je tzv. funkcia nákladov a $F(R) = f(f_1(c_1(R)),\ \ldots\ , f_n(c_n(R)))$) sú niektorého z nasledujúcich troch typov.

A) Súčet hodnôt funkcie nákladov za všetky všetky úlohy v rozvrhu (t.j. f je suma): $f = \sum_{i=1}^{n} f_i(c_i(R))$. Kritériá tohoto typu sa označujú veľkým písmenom (napr. C pre súčet časov ukončenia).

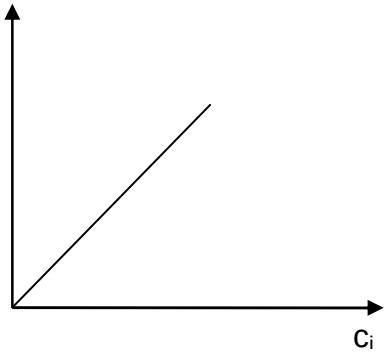
B) Maximálna hodnota spomedzi hodnôt funkcie nákladov za všetky úlohy v rozvrhu (f je maximum): $f = \max_i f_i(c_i(R))$. Kritériá tohoto typu sa označujú veľkým písmenom s indexom max (napr. C_{max} pre dĺžku rozvrhu – čas ukončenia poslednej úlohy, alebo L_{max} – najväčšie oneskorenie).

C) Priemerná hodnota spomedzi hodnôt funkcie nákladov za všetky úlohy v rozvrhu (f je aritmetický priemer): $f = \dfrac{\sum_{i=1}^{n} f_i(c_i(R))}{n}$, prípadne ak sú zadané priority (váhy úloh), potom sa používa vážený priemer: $f = \dfrac{\sum_{i=1}^{n} w_i f_i(c_i(R))}{\sum_{i=1}^{n} w_i}$. Kritériá tohoto typu sa označujú veľkým písmenom s pruhom (napr. $\overline{C}$ pre priemerný čas ukončenia, alebo $\overline{F}$ pre priemerný čas pobytu vo výrobe), alebo s indexom w v prípade váženého priemeru (napr. L_w pre vážený priemer oneskorení).

Najčastejšie používané funkcie nákladov sú nasledovné.

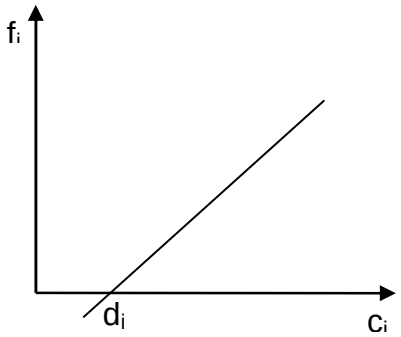
C (completion time – čas ukončenia)

$$f_i = c_i(R)$$
$$f_i = w_i c_i(R)$$



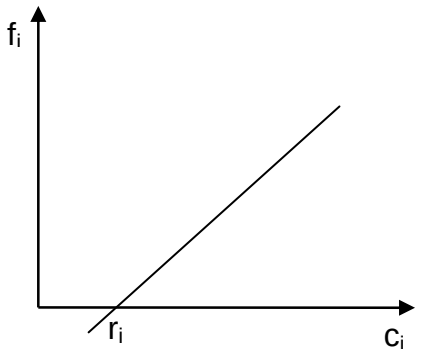
L (lateness time – oneskorenie)

$$l_i = c_i - d_i$$
$$l_i = w_i(c_i - d_i)$$



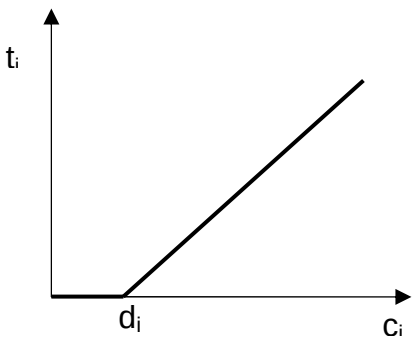
F (flow time – dĺžka spracovania)

$$f_i = c_i - r_i$$
$$f_i = w_i(c_i - r_i)$$



T (tardeness time – dĺžka omeškania)

$$f_i = \max(0, \quad w_i(c_i - d_i))$$



n_T (počet omeškaných úloh) $f_i = sign(\max(0, \quad w_i(c_i - d_i)))$

Skúsme teraz vypočítať hodnoty kriteriálnych funkcií pre rozvrhy *R1* a *R2* z predchádzajúceho príkladu. Výsledky a porovnanie hodnôt rôznych kritériálych funkcií pre oba rozvrhy je zhrnuté na Obr. 23. Najlepšie hodnoty podľa daného kritéria sú označené hviezdičkou.

	R_1	R_2
$C(R_i)$	[3, 4, 5, 6, 1, 8, 9, 8]	[4, 4, 5, 6, 6, 8, 8, 8]
l_i	[-2, 0, 0, 3, -6, 2, 0, -4]	[-1, 0, 0, 3, -1, 2, -1, -4]
C_{max}	9	8 *
$\overline{C}$	5,5 *	6,1
C	44 *	49
L_{max}	3 *	3 *
$\overline{L}$	-1 *	0,25
L	-7 *	-2
F_{max}	9	8 *
$\overline{F}$	5,5 *	6,1
F	44 *	49
$\overline{E}$	1,5 *	1
$\overline{n_T}$	2 *	2 *

Obr. 23 Výpočet rôznych kriteriálnych funkcií pre rozvrhy *R1* a *R2* z predchádzajúceho príkladu

Príklad

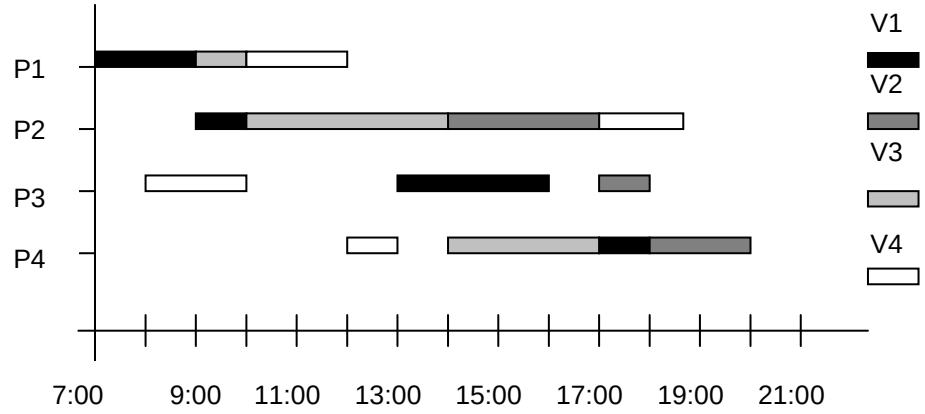
Úlohou je navrhnúť rozvrh výroby pre štyri rôzne výrobky V1 až V4. Každý z týchto výrobkov má presne definované poradie operácií, ktoré sa realizujú na niektorých, alebo na všetkých pracoviskách P1 až P4. Na každom z týchto pracovísk možno spracovávať len jeden výrobok v danom čase a každé z pracovísk je k dispozícii len v určitom časovom intervale. Každý výrobok musí prejsť jednotlivými operáciami (každá operácia na zadanom pracovisku) presne v zadanom poradí a časy trvania jednotlivých operácií pre rôzne výrobky môžu byť rôzne. Spracovávanie výrobku na danom pracovisku nesmie byť prerušené (jedna operácia nesmie byť prerušená inou) a každý výrobok má špecifikovaný najneskorší čas, kedy musí byť hotový. Všetky požiadavky pre túto úlohu sú zhrnuté v Tab. 4.

Výrobok	Najskorší čas začiatku výroby	Poradie operácií na jednotlivých pracoviskách a ich trvanie	Najneskorší čas ukončenia výroby
V1	7:00	P1(2),P2(1),P3(3),P4(1)	20:00
V2	9:00	P2(3),P3(1),P4(2)	18:00
V3	9:00	P1(1),P2(4),P4(3)	21:00
V4	8:00	P3(2),P1(2),P4(1),P2(2)	19:00

Tab. 4 Výrobné požiadavky pre príklad úlohy typu job-shop

Na Obr. 24 je ukážka rozvrhu, ktorý spĺňa všetky ohraničenia dané v tomto príklade. Avšak mnohé úlohy vyžadujú optimalizáciu, t.j. nájdenie takého rozvrhu, ktorý minimalizuje (prípadne maximalizuje) nejaké kritérium. Napríklad nájsť taký rozvrh, v ktorom by boli všetky výrobky hotové tak skoro, ako je to len možné.

V tejto kapitole budú presne zadefinované jednotlivé typy rozvrhovacích úloh a vybrané metódy, ktorými je možné tieto úlohy riešiť. V závere kapitoly je uvedené zhrnutie a porovnanie širšieho spektra metód používaných na riešenie rozvrhovacích úloh.



Obr. 24 Príklad prípustného rozvrhu pre úlohu danú v príklade

Riešenie

Pre riešenie tejto úlohy použime jej reprezentáciu ako úlohy s ohraničeniami (viď. aj časť 2.3, resp. podrobne v [Mach, Paralič 2000]), t.j. potrebujeme ju definovať pomocou množiny premenných, ich domén a množiny ohraničení medzi týmito premennými. Úlohou je nájsť také priradenie hodnôt premenným, aby boli splnené všetky ohraničenia.

Ako premenné možno zvoliť v našom príklade časy kedy sa začne spracovávať daný výrobok na danom pracovisku (začiatok vykonávania jednej operácie). Takže pre zadaný príklad to znamená celkovo 14 premenných (po 4 pre *V1*, *V4* a po 3 pre *V2*, *V3*).

Nech T_{XY} reprezentuje začiatok spracovávania výrobku V_X na pracovisku P_Y a D_{XY} trvanie tohoto spracovania. Potom jednotlivé typy ohraničení z definície úlohy možno reprezentovať nasledovne:

- Požiadavku najskoršieho času započatia výroby napríklad pre výrobok *V1* možno reprezentovať nerovnicou $T_{11} \geq 7$.
- Požiadavku presného poradia pracovísk pri spracovávaní daného výrobku možno vyjadriť opäť ako nerovnice, kde sa zohľadňuje trvanie jednotlivých operácií. Napr. operácia s časom začiatku T_{11} trvá 2 hodiny a T_{12} môže začať až po skončení T_{11}, takže $T_{11} + 2 \leq T_{12}$.
- Aby sa zabezpečilo, že v danom okamžiku môže na jednom pracovisku prebiehať len jedna operácia, musí sa definovať množina dvojíc disjunktných ohraničení, z ktorých zakaždým bude platiť práve jedno. Vo všeobecnosti pre pracovisko P_Y a výrobky V_X a V_{X^*} ktoré majú byť na tomto pracovisku spracovávané musí platiť práve jedna z nasledujúcich dvoch nerovníc.

$$\forall\ X^* \neq X : T_{XY} + D_{XY} \leq T_{X^*Y} \text{ alebo } T_{X^*Y} + D_{X^*Y} \leq T_{XY}$$

Ak platí prvá nerovnica, potom pracovisko P_Y spracováva výrobok V_X pred výrobkom V_{X*}. Naopak ak platí druhá nerovnica, potom pracovisko P_Y spracováva výrobok V_{X*} pred výrobkom V_X.

Disjunktné ohraničenia sú hlavným zdrojom komplexnosti rozvrhovacích úloh, z ktorých sú väčšina NP-úplné problémy.

6.2 Rozvrhovanie na paralelných procesoroch (strojoch)

V prípade že na riešenie jednotlivých úloh je možné použíť ľubovoľný z dostupných procesorov (strojov), hovoríme o úlohách rozvrhovania na paralelných procesoroch (strojoch).

V tejto kategórií úloh rozlišujeme dva základné prípady, a síce tzv. jednoprocesorové úlohy (alebo jednostupňová výroba) v prípade, že máme k dispozícii iba jeden procesor (stroj) – tieto úlohy sú podrobnejšie popísané v nasledujúcej časti 6.2.1.

Druhú skupinu tvoria viacprocesorové úlohy (resp. viacstupňová výroba), kedy máme k dispozícii na rozvrhovanie viacero procesorov (strojov). Týmto úlohám je venovaná časť 6.2.2.

6.2.1 Rozvrhovanie na jednom procesore (stroji)

Na prvý pohľad by sa mohlo zdať, že rozvrhovanie úloh na jednom procesore nie je zložitá úloha. Žiaľ, nie je tomu tak. Stačí si uvedomiť, že ak máme rozvrhnúť n úloh na jednom procesore, máme $n!$ možností v prípade rozvrhovania bez prerušenia, čo svedčí o hrozbe kombinatorickej explózie.

Existuje niekoľko špeciálnych prípadov, ktoré sa dajú riešiť triviálne, alebo existuje pre ne dobrý polynomiálny algoritmus. Väčšina týchto úloh však patrí do skupiny nepolynomiálnych úloh.

Ak pripustíme prerušenie úloh počas ich spracovania (angl. preemption), potom úloha rozvrhovania na jednom procesore s cieľom optimalizovať najviac oneskorenú úlohu (kritérium L_{max}) je riešiteľná veľmi exektívne Jacksonovým algoritmom.

Jacksonov algoritmus:

1. Vždy aktivuj úlohu s najskoršou dobou ukončenia.
2. Akonáhle začne byť úloha T_i pripravená a procesor je obsadený úlohou T_j, pozastav úlohu T_j v prospech úlohy T_i práve vtedy, ak čas ukončenia i-tej úlohy je skorší ako čas ukončenia j-tej úlohy, inak ponechaj bežať úlohu T_j.
3. Pokračuje krokom 1, kým nie sú dokončené všetky úlohy.

Príklad

Je zadaných šesť úloh (A až E), ktoré je potrebné rozvrhnúť na jeden procesor, pričom je povolené prerušenie spracovania úloh a kritériom L_{max}.

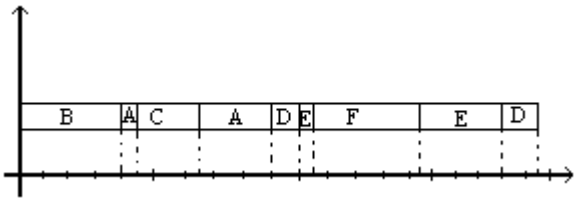
Časy trvania, požadované časy ukončenia a časy, kedy sú jednotlivé úlohy pripravené na spracovanie uvádza Tab. 5.

Úloha	t_i	r_i	d_i
A	6	4	32
B	8	0	27
C	4	9	22
D	5	15	43
E	8	20	38
F	8	21	36

Tab. 5 Zadanie úlohy rozvrhovania na jednom procesore, s prerušením

Riešenie

Aplikovaním Jacksonovho algoritmu získame rozvrh, ktorého Ganttov diagram je zobrazený na Obr. 25.



Obr. 25 Optimálny rozvrh pre zadanú úlohu a kritérium L_{max}

Presné hodnoty časov ukončenia (c_i) a časov oneskorenia oproti zadaným požadovaným časom ukončenia sú uvedené v Tab. 6.

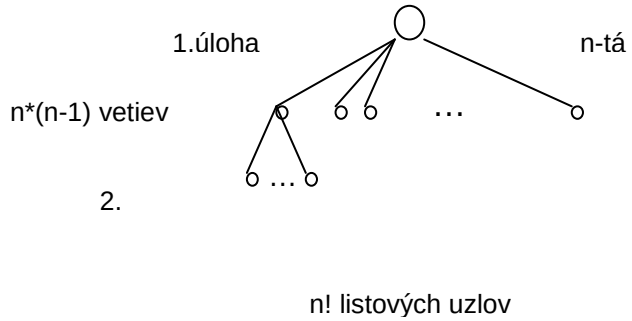
Úloha	t_i	r_i	d_i	c_i	l_i
A	6	4	32	18	-14
B	8	0	27	8	-19
C	4	9	22	13	-9
D	5	15	43	39	-4
E	8	20	38	36	-2
F	8	21	36	29	-7
					L_{max} = -2

Tab. 6 Riešenie úlohy rozvrhovania na jednom procesore s prerušením, najdené pomocou Jacksonovho algoritmu

V prípade že ide o úlohu rozvrhovania na jednom procesore ***bez prerušenia,*** je to vo všeobecnosti zložitejšia úloha, nakoľko má permutačný charakter (*n!* možných rozvrhov), ktorú až na špeciálne prípady nemožno riešiť v polynomiálnom čase. Niektoré špeciálne prípady nasledujú.

1. Ak máme zadaných n úloh T_i $(i = 1, ..., n)$, pričom všetky sú k dispozícii hneď od začiatku, t.j. $r_i = 0$ $(i = 1, ..., n)$, nie sú zadané žiadne precedenčné ohraničenia, žiadne priority a ani požadované časy ukončenia, potom:
   a. Z hľadiska kriteriálnej funkcie C_{max} sú všetky rozvrhy rovnako dobré
   b. Z hľadiska kriteriálnej funkcie C je optimálne usporiadanie úloh podľa neklesajúcej postupnosti ich dĺžok trvania, tj.: $t_{(1)} \leq t_{(2)} \leq ... \leq t_{(n)}$
2. Ak máme zadaných n úloh T_i $(i = 1, ..., n)$, pričom všetky sú k dispozícii hneď od začiatku, t.j. $r_i = 0$ $(i = 1, ..., n)$, nie sú zadané žiadne precedenčné ohraničenia ani požadované časy ukončenia, ale úlohy majú rôzne priority w_i $(i = 1, ..., n)$, potom:
   c. Z hľadiska kriteriálnej funkcie C_w je optimálne usporiadanie úloh podľa nerastúcej postupnosti ich priorít, tj.: $w_{(1)} \geq w_{(2)} \geq ... \geq w_{(n)}$
3. Ak máme zadaných n úloh T_i $(i = 1, ..., n)$, pričom všetky sú k dispozícii hneď od začiatku, t.j. $r_i = 0$ $(i = 1, ..., n)$, nie sú zadané žiadne precedenčné ohraničenia ani priority, ale sú zadané požadované časy ukončenia úloh d_i $(i = 1, ..., n)$, potom:
   d. Z hľadiska kriteriálnej funkcie L_{max} existuje viacero heuristík, napr. Moorov algoritmus vychádzajúci z neklesajúcej postupnosti požadovaných časov ukončenia úloh, t.j. $d(1) \leq d(2) \leq ... \leq d(n)$. Ďalšie používané heuristiky: Smithov algoritmus a Lawlerov algoritmus [Dupaľ, Brezina 2005].

Všetky ostatné úlohy vedú na permutačné rozvrhy a je možné ich riešiť napríklad metódou vetvenia a medzí, ktorá sa snaží efektívne prehľadať priestor prehľadávania, ktorého štruktúra je znázornená na Obr. 26.



Obr. 26 Štruktúra priestoru prehľadávania pre riešenie úoh rozvrhovania na jednom procesore bez prerušenia

6.2.2 Rozvrhovanie na viacerých procesoroch (strojoch)

Pri rozvrhovaní na viacerých paralelných procesoroch sa využívajú rôzne

heuristiky, vychádzajúce z nejakého usporiadania úloh. Jednou z takýchto heuristík je aj LPT (Longest Processing Time), ďalšie heuristiky sú uvedené v časti 6.5.

```
 Heuristika LPT:


Vytvor zoznam úloh t(1)≥ t(2)≥ ... ≥ t(n)
    for j=1 to m Sj=0;
    j := 1
    repeat
       urči také k, že Sk=min{Si}
                       1≤i≤m
       priraď úlohu Tj (prvá v aktuálnom zozname) na procesor k
       Sk := Sk+tj;
       j := j+1;
    until j=n;
end;
```

Pre heuristiku LPT bola dokázaná najväčšia možná chyba, ktorej sa môže dopustiť tento algoritmus oproti optimálnemu rozvrhu (viď. aj nasledujúci príklad):

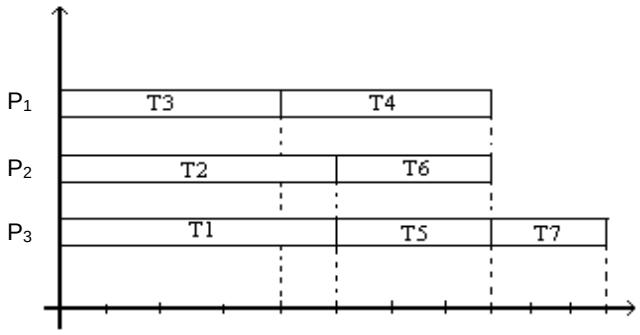
$Q_{LPT} = \frac{4}{3} - \frac{1}{3m}$, t.j. napr. pre tri procesory $Q_{LPT} = \frac{4}{3} - \frac{1}{9} = \frac{12-1}{9} = \frac{11}{9}$

Príklad

Je zadaných sedem úloh (*T1* až *T7*), ktoré je potrebné rozvrhnúť na tri paralelné procesory, pričom dĺžky trvania jednotlivých úloh sú v poradí nasledovné: [5, 5, 4, 4, 3, 3, 3].

Riešenie

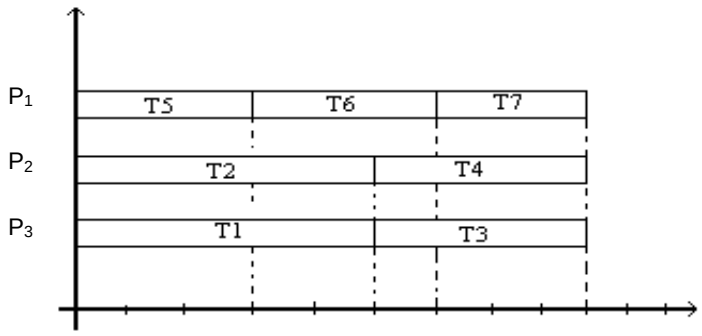
Použitím heuristiky LPT možno zostrojiť rozvrh R1, ktorý je zobrazený na Obr. 27.



Obr. 27 Rozvrh pre zadanú úlohu pre paralelné proceosry, zostrojený heuristikou LPT

Tento rozvrh je vzdialený od optimálneho práve najhorším možným spôsobom, t.j. v prípade troch procesorov je práve $Q_{LPT} = \frac{11}{9}$.

Optimálny rozvrh v zmysle kritéria C_{max} je znázornený na Obr. 28.



Obr. 28 Optimálny rozvrh pre zadanú úlohu podľa kritéria C_{max}

6.3 Rozvrhovanie na dedikovaných procesoroch (strojoch)

V prípade že na riešenie jednotlivých úloh je možné použíť iba špecializované (dedikované) procesory (stroje), hovoríme o úlohách rozvrhovania na dedikovaných procesoroch (strojoch).

V tomto prípade sa úlohy zvyknú rozdeľovať do skupín, tzv. zákaziek (angl. jobs), pričom v rámci jednej zákazky sa jednotlivé úlohy vykonávajú na navzájom rôznych špecializovaných procesoroch (strojoch). Od predpísaného poradia a počtu úloh v zákazkách potom závisí, o ktorý z troch hlavných typov úloh na dedikovaných procesoroch (strojoch) sa jedná. Predpokladajme, že v rozvrhovacej úlohe je definovaných k zákaziek, ktoré označíme $J_k = [T_{1,k}, \ldots, T_{nk,k}]$, kde n_i je počet úloh v i-tej zákazke.

Základné charakteristiky troch typov rozvrhovacích úloh na dedikovaných procesoroch (strojoch) sú zhrnuté v Tab. 7.

Typ rozvrhovacej úlohy	Počet úloh vrámci zákaziek n_k	Poradie úloh v rámci zákaziek J_k
Open shop	***Rovnaké*** pre všetky zákazky J_k	***Ľubovoľné***
Flow shop	***Rovnaké*** pre všetky zákazky J_k	***Pevne dané, rovnaké*** pre všetky zákazky J_k
Job shop	***Rôzne*** pre jednotlivé zákazky J_k	***Pevne dané, rôzne*** pre jednotlivé zákazky J_k

Tab. 7 Prehľad základných vlastností jednotlivých typov rozvrhovacích úloh na dedikovaných procesoroch

6.3.1 Úlohy typu flow shop

Vo všeobecnosti ide o kombinatorickú optimalizáciu, existuje len niekoľko málo prípadov riešiteľných v polynomiálnom čase (podrobnejšie napr. v [Blazewicz et al. 1996]).

Jedným z takýchto špeciálnych prípadov je flow shop na dvoch procesoroch (s ľubovoľným počtom zákaziek J, všetky úlohy sú k dispozícii v čase 0), ktorý sa rieši Johnsonovým algoritmom pre nájdenie optimálneho rozvrhu podľa kritéria C_{max}.

Johnsonov algoritmus pre job shop na dvoch procesoroch pre ľubovoľný počet zákaziek, tj. každá zákazka J_i je zložená z dvoch úloh, prvá úloha musí byť vykonaná na prvom procesore a trvá t_{1i}, druhá úloha zase na druhom procesore v trvaní t_{2i}.

Johnsonov algoritmus:

1. Z množiny všetkých zákaziek $J = \{J_1, \ldots, J_n\}$ vytvoríme dva zoznamy: $L_1 = \{J_i \mid t_{1i} \leq t_{2i}\}$ a $L_2 = J - L_1$
2. Zoznam L_1 usporiadame podľa neklesajúcich procesných časov t_{1i} a zoznam L_2 podľa nerastúcich časov t_{2i}
3. Optimálny rozvrh v zmysle kritéria C_{max} je tvorený zreťazením zoznamov L_1 a L_2

Príklad

Majme 2 procesory a 7 zákaziek J_i (i = 1 .. 7) s procesnými časmi danými v Tab. 8.

J_i	1	2	3	4	5	6	7
P_{1i}	4	6	1	4	5	7	9
P_{2i}	3	8	3	5	2	9	5

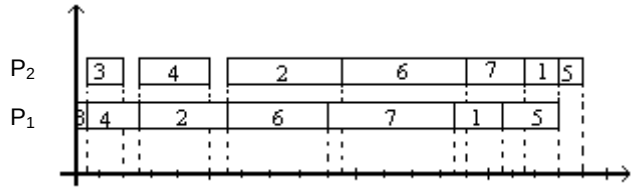
Tab. 8 Zadanie úlohy typu flow shop – na dvoch procesoroch, sedem úloh v zákazke

Riešenie

Aplikujeme Johnsonov algoritmus, podľa ktorého dostávame:

1. $L_1 = \{J_2, J_3, J_4, J_6\} \Rightarrow L_2 = \{J_1, J_5, J_7\}$
2. Usporiadané: $L_1 = \langle J_3, J_4, J_2, J_6 \rangle$, $L_2 = \langle J_7, J_1, J_5 \rangle$
3. Takže výsledný optimálny rozvrh podľa C_{max} zodpovedá zreťazeniu L_1 a L_2
4. $R = \langle J_3, J_4, J_2, J_6, J_7, J_1, J_5 \rangle$

Výsledný optimálny rozvrh vzhľadom na kritérium C_{max} je zobrazený na Obr. 29.



Obr. 29 Optimálny rozvrh pre riešenie vyššie zadanej úlohy typu flow shop

Johnsonov algoritmus sa dá použiť aj pre úlohu flow shop na 3 procesoroch (t.j. flow shop m = 3, n = ľubovoľné), ale len ak platia nasledujúce dve obmedzenia.

$$\min_i\{t_{1i}\} \geq \max\{t_{2i}\}$$

$$\min_i\{t_{3i}\} \geq \max\{t_{2i}\}$$

6.3.2 Úlohy typu open shop

Každé riešenie úlohy typu flow shop je aj riešením úlohy open shop. Otázkou je, či neexistuje aj lepší rozvrh. Vo všeobecnosti ide opäť väčšinou o nepolynomiálne úlohy. Niektoré algoritmy špeciálne na riešenie úloh tohoto typu možno nájsť napr. v [Blazewicz et al. 1996].

6.3.3 Úlohy typu job shop

Ide o relatívne najzložitejšie rozvrhovacie úlohy. Počet úloh v jednotlivých zákazkách môže byť rôzny, poradie úloh v rámci zákaziek je pevne dané, ale rôzne pre jednotlivé zákazky. Zo špeciálnych prípadov spomenieme úlohu job shop s dvoma zákazkami, pri ľubovoľnom počte úloh v zákazkách. Táto úloha sa dá riešiť graficky. Postup je nasledovný:

Algoritmus na riešenie úloh typu job shop pre dve zákazky a ľubovoľný počet úloh v zákazkách:

1. Na vodorovnú os vynesieme časy spracovania úloh v rámci prvej zákazky v predpísanom poradí podľa jednotlivých procesorov.

2. Na zvislú os vynesieme časy spracovania úloh v rámci druhej zákazky v predpísanom poradí podľa jednotlivých procesorov.

3. Vyšrafujeme tzv. „zakázané oblasti“, t.j. oblasti v ktorých na vodorovnej aj zvislej osi patria tomu istému procesoru.

4. Rozvrh znázorňujeme tzv. „pracovnou čiarou“ . ide o lomenú čiaru pozostávajúcu z troch typov úsekov:
   - Úseky pod 45° uhlom zodpovedajú paralelnému spracovávaniu oboch úloh
   - Vodorovný úsek zodpovedá spracovávaniu prvej zákazky, zatiaľ čo druhá čaká na uvoľnenie procesora.

- Zvislý úsek znamená spracovávanie druhej zákazky, zatiaľ čo prvá čaká na uvoľnenie procesora.

5. Najkratšia pracovná čiara potom zodpovedá optimálnemu rozvrhu.

Príklad

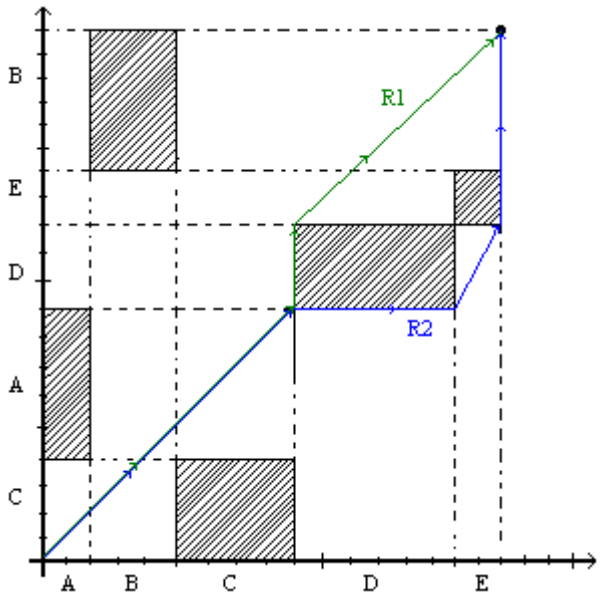
Majme 2 zákazky J_1 a J_2 s úlohami s trvaním a poradím daným procesnými časmi danými v Tab. 9.

J_1	Poradie	A	B	C	D	E	$\sum$
	Trvanie	2	3	4	6	2	17
J_2	Poradie	C	A	D	E	B	$\sum$
	Trvanie	4	5	3	2	6	20

Tab. 9 Zadanie úlohy typu job shop – dve zákazky, päť úloh v zákazke

Riešenie

Aplikáciou vyššie popísanej grafickej metódy dostaneme nákres zobrazený na Obr. 30.



Obr. 30 Riešenie danej úlohy typu job shop pre dve zákazky grafickou metódou

6.4 Prehľad ďalších metód na riešenie úloh rozvrhovania

V tejto časti budú stručne predstavené viaceré metódy, ktoré boli úspešne použité na riešenie niektorých typov úloh rozvrhovania. Pre každú z metód bude stručne popísaný postup, ako sa pri riešení úloh rozvrhovania používa, jej výhody a nevýhody. V závere sú potom všetky prezentované metódy prehľadne porovnané vo forme tabuľky.

6.4.1 Lineárne programovanie

Aby bolo možné použiť lineárne programovanie, je nutné najprv rozvrhovaciu úlohu reprezentovať spôsobom opísaným na príklade v úvode tejto kapitoly 6. Reprezentácia tejto úlohy ako úlohy s ohraničeniami (CSP) v zásade vyhovuje aj požiadavkám pre lineárne programovanie. Takže len stručne zrekapitulujem.

Premenné označujú začiatočné časy jednotlivých úloh. T_{XY} označuje začiatočný čas operácie na výrobku V_X vykonávanej na pracovisku Py a D_{XY} je trvanie tejto operácie. Potom platia všetky tri skupiny ohraničení uvedených v úvode tejto kapitoly 6. Avšak posledná skupina ohraničení sú disjunktné, a síce pre každú dvojicu operácií zdieľajúcich ten istý zdroj (rovnaké pracovisko) máme disjunktné ohraničenie s dvoma alternatívami, z ktorých len jedna bude platiť vo výslednom rozvrhu.

Tieto disjunktné ohraničenia sa obyčajne spracúvajú vymenovaním všetkých kombinácií nerovníc. Takto dostaneme príslušný počet konjunktívnych množín nerovníc, z ktorých každá sa musí riešiť samostatne.

K úplnosti formulácie úlohy ešte chýba optimalizačná funkcia. Nakoľko táto nebola v zadaní požadovaná (chcem len nájsť rozvrh spĺňajúci všetky zadané ohraničenia), stačí zvoliť nejakú funkciu, napr. $\sum T_{XY}$.

Simplexová metóda a modifikovaná simplexová metóda sú najčastejšie používané algoritmy na riešenie úloh lineárneho programovania. Dôležitou vlastnosťou lineárneho programovania je, že ak riešenie existuje, vždy nájde optimálne riešenie úlohy. Avšak v prípade veľkého počtu disjunktných ohraničení, čo je dosť časté pri úlohách rozvrhovania, aplikovateľnosť tejto metódy je obmedzená vzhľadom na kombinatorickú explóziu.

6.4.2 Metóda vetvenia a medzí

Ide o pomerne dobre známy algoritmus z operačného výskumu pre účely optimalizácie (podrobne viď. časť 2.2.1). Metóda vetvenia a medzí je úplná metóda prehľadávania. V zásade ide o prehľadávanie do hĺbky plus použitie vyhodnocovacej funkcie pre orezanie priestoru prehľadávania.

Metóda vetvenia a medzí prehľadáva priestor stavov (každý stav predstavuje uzol v priestore prehľadávania). V rozvrhovaní môže byť stavom priradenie hodnôt určitej podmnožine premenných. Napr. premennými môžu byť počiatočné časy operácií z príkladu uvedeného na začiatku tejto kapitoly 6. Funkcia susednosti definuje štruktúru priestoru prehľadávania (t.j. ktoré stavy sú priamo prístupné z ktorých - uzol reprezentujúci stav sa takto rozvetvuje). Napr. potomok daného stavu môže byť získaný priradením hodnoty premennej, ktorej ešte nebola priradená hodnota v rodičovských uzloch.

Postup začína koreňovým uzlom, v ktorom nie je žiadne priradenie hodnôt premenným. Tento uzol sa ďalej vetví (pre každú možnú hodnotu prvej zvolenej premennej jedna vetva). Algoritmus v každom kroku preskúma jeden uzol (jedného potomka aktuálneho uzla, resp. jeho súrodenca, ak už preskúmal všetkých potomkov), pričom postupuje smerom do hĺbky, až kým nepreskúma všetky vetvy. Naviac si algoritmus metódy vetvenia a medzí pamätá doposiaľ najlepšie riešenie a skôr než začne skúmať ďalší uzol (t.j.

všetkých jeho potomkov definovaných funkciou susednosti), využije doménové znalosti vo forme vyhodnocovacej funkcie, aby odhadol hodnotu optimálneho riešenia pod týmto uzlom. Ak táto odhadnutá hodnota je horšia než doposiaľ najlepšie nájdené riešenie, zvolený uzol nebude preskúmaný. Pre korektnosť tejto metódy je preto nevyhnutné, aby vyhodnocovacia funkcia nikdy nepodhodnotila (nenadhodnotila) skutočné optimálne hodnoty v maximalizačnej (minimalizačnej) úlohe.

Metódu vetvenia a medzí možno použiť napr. aj v kombinácii s lineárnym programovaním popísaným v predchádzajúcom bode. Stavom je v takomto prípade konjunkcia nerovníc. Potomok uzla k tejto množine pridáva ďalšiu nerovnicu. Listový uzol obsahuje kompletnú konjunkciu nerovníc, ktorých splnenie definuje úplný rozvrh.

Efektívnosť tejto metódy je veľmi ovplyvňovaná:

- Kvalitou vyhodnocovacej funkcie, ktorá robí odhad optimálneho riešenia pod daným uzlom (čím presnejší odhad, tým viac vetiev je možné orezať).
- Ďalším dôležitým faktorom určujúcim počet stavov, ktorých prehľadávanie sa môže orezať, je poradie v ktorom sú vetvy prehľadávané (čím skôr sa nájde kvalitnejšie riešenie, tým účinnejšie orezávanie).
- A nakoniec nemenej dôležitým je aj spôsob reprezentácie úlohy, ktorý determinuje veľkosť priestoru prehľadávania (viď. príklad o rozvrhovaní rezania kusov dreva podrobne popísaný v [Dincbas et al. 1988] kde sa veľkosť priestoru prehľadávania znížila z pôvodných 4^{72} (čo je cca 10^{43}) na 7^{42} (cca 10^{7}) len zmenou spôsobu reprezentácie úlohy).

6.4.3 Spĺňanie ohraničení

Tento smer v umelej inteligencii priniesol množstvo algoritmov predovšetkým na riešenie úloh splniteľnosti [Dechter 1992], [Paralič, Sabol 1995] (t.j. pre úlohy rozvrhovania to znamená nájdenie nejakého riešenia spĺňajúceho všetky ohraničenia). Tieto algoritmy sú určené na všeobecne formulované tzv. konečné úlohy spĺňania ohraničení (z anglického finite constraint satisfaction problem - FCSP).

FCSP zahrňuje množinu premenných, z ktorých každá má konečnú doménu (množinu prípustných hodnôt) a množinu ohraničení, ktoré rozličným spôsobom obmedzujú hodnoty ktoré môžu premenné nadobúdať [Paralič, Sabol 1995]. Úlohou je priradiť každej premennej hodnotu z jej domény tak, aby boli splnené všetky ohraničenia. Ako už bolo spomínané pre tento typ úloh bolo vyvinutých množstvo algoritmov ktoré na tomto mieste nebudem uvádzať. Podrobne spracovaný prehľad týchto metód možno nájsť v [Mach, Paralič 2000] [Paralič, Sabol 1995].

Na rozdiel od lineárneho programovania pri spĺňaní ohraničení nemusia byť len numerické premenné, môžu to byť aj enumeračné premenné s ľubovoľnou konečnou množinou prípustných symbolov. Takisto nie je žiadne obmedzenie na typ prípustných ohraničení. Táto flexibilita znamená, že spĺňanie ohraničení má široké spektrum použitia.

Metódy z tejto skupiny sú využívané aj vnútri CLP systémov. Podrobne

možno nájsť popis používaných algoritmov a techník v [Paralič 1995].

Vráťme sa opäť pre účely porovnania k príkladu z úvodu tejto kapitoly 6. Existuje viacero spôsobov ako ho vyjadriť formou CSP. Jedným z nich je aj spôsob popísaný vyššie, t.j. použiť numerické premenné pre počiatočné časy operácií. Počiatočné domény budú dané dostupnosťou zdroja, na ktorý je tá ktorá operácia viazaná. Napr. operácia T_{22} môže byť inicializovaná doménou $\langle 7..20 \rangle$ (čo znamená interval celých čísel od 7 po 20) čo je dostupnosť na túto operáciu požadovaného pracoviska $P2$ od 7. do 20. hodiny. Nakoľko však operácia T_{22} trvá 3 hodiny, jej počiatočný čas musí byť z intervalu $\langle 7..17 \rangle$. Okrem toho ďalšie ohraničenie udáva, že najskorší možný čas pre začiatok výroby výrobku $V2$ je 9 hodín a najpozdnejší termín ukončenia jeho výroby 18 hodín, takže doména T_{22} sa zúži na $\langle 9..15 \rangle$.

Popísaný postup zodpovedá algoritmu uzlovej konzistentnosti [Mach, Paralič 2000], ktorý zužuje doménu premennej na základe unárnych (t.j. týkajúcich sa len tejto premennej) ohraničení, ktoré musí táto premenná spĺňať.

Ohraničenia v tejto úlohe môžu byť definované ako funkcie, ktoré vrátia hodnotu “pravda”, ak aktuálna kombinácia priradených hodnôt spĺňa dané ohraničenie a “nepravda” v opačnom prípade. Tieto funkcie spravidla opäť šíria ohraničenia do určitej miery. Napríklad ak sú známe hodnoty všetkých premenných okrem jednej, zredukuje sa jej doména tak, že sa z nej vyradia všetky neprípustné hodnoty (tzv. forward checking - dopredná kontrola). Vo všeobecnosti čím viac času sa obetuje na propagáciu ohraničení, tým menší priestor prehľadávania. Tu je ale opäť nutné hľadať rozumný kompromis medzi mierou propagácie a časom stráveným prehľadávaním. Oba procesy sú obvykle úzko spojené (detaily viď. [Mach, Paralič 2000]). Aj táto skupina metód naráža na kombinatorickú explóziu.

Techniky spĺňania ohraničení sú pružnejšie než lineárne programovanie a majú širšiu oblasť použiteľnosti. Avšak väčšina metód nerieši optimalizačný problém. Niektoré metódy môžu však byť integrované do metódy vetvenia a medzí na dosiahnutie lepšej efektívnosti.

Efektívnosť výslednej aplikácie opäť veľmi ovplyvní:

- spôsob formulácie problému (vo všeobecnosti je zložitosť priamo úmerná súčinu počtu premenných a veľkosti ich domén, takže čím menej premenných a čím menšie sú ich domény, tým lepšie),
- poradie premenných v akom im budú priradzované hodnoty v priebehu prehľadávania,
- poradie hodnôt z aktuálnej domény, v akom budú brané ako alternatívne hodnoty pre danú premennú.

6.4.4 Hill climbing

Touto metódou začína popis stochastických metód, alebo metód lokálneho prehľadávania (okrem hill climbing sem patrí aj simulované žíhanie a prehľadávanie tabu). Hill climbing [Tsang 1993] je ich najjednoduchšou verziou, ale princíp je u nich rovnaký. Všetky sú totiž používané pre optimalizačné úlohy, kde je akceptovateľné aj suboptimálne riešenie. Tieto

suboptimálne riešenia sú v praxi dosť často akceptovateľné (najmä ak priestor prehľadávanie je priveľký pre úplné metódy prehľadávania uvedené v predchádzajúcich podkapitolách).

Hill climbing vyžaduje funkciu susednosti, ktorá mapuje každý stav na skupinu ďalších - "susedných" stavov v priestore prehľadávania. Kvalita definície tejto funkcie (to si vyžaduje doménovo závislé znalosti) výrazne ovplyvňuje efektívnosť algoritmu.

Základný postup u hill climbing je veľmi jednoduchý. Počínajúc z náhodne (alebo heuristicky) generovaného stavu sa prejde do takého susedného stavu, ktorý je "lepší" vzhľadom na optimalizačnú funkciu. Tento proces sa opakuje až do chvíle, kým žiadny ďalší lepší prechod už neexistuje. Heuristika, ktorá vyberá z lepších susedných stavov môže byť rôzna (napr. vyber ľubovoľný, alebo vyber najlepší).

Hill climbing je dôležitou metódou pre riešenie úloh, u ktorých použitie úplných metód prehľadávania (viď. predchádzajúce podkapitoly) už zlyháva v dôsledku kombinatorickej explózie. Obyčajne nie je zložité vyvinúť stratégiu pre hill climbing, ale nájsť dobré riešenia (blízke optimálnemu) je vždy dosť ťažké. Hlavným nedostatkom hill climbing je jeho náchylnosť uviaznuť v lokálnych optimách prípadne v oblastiach, kde sa nemení kvalita susedných riešení (rovinkách).

6.4.5 Simulované žíhanie

Simulované žíhanie [Kirkpatrick et al. 1983], [Tsang 1995] je rozšírením metódy hill climbing s cieľom vyviaznuť z lokálnych optím. Znamená to, že je tu aj určitá pravdepodobnosť, že smer postupu od jedného stavu k nasledujúcemu už nemusí byť len jedným smerom (od horšieho k lepšiemu), ale s určitou pravdepodobnosťou je možný aj ľubovoľný iný prechod.

Metóda vznikla v prvej polovici osemdesiatych rokov [Kirkpatrick et al. 1983]. Bola inšpirovaná procesom eliminácie defektov kryštálovej mriežky kryštálov ich ohriatím s nasledovným pomalým ochladzovaním na nízku teplotu.

Pri tomto algoritme teplota vystupuje ako riadiaci parameter, ktorý určuje, ktoré nové stavy sú akceptovateľné a ktoré nie. Vychádzajúc z preddefinovanej počiatočnej teploty, ktorá sa postupne znižuje (podľa tzv. plánu ochladzovania), majú aj slabšie susedné riešenia šancu byť vybrané. Pravdepodobnosť vybratia horšieho riešenia (vzhľadom na optimalizačnú funkciu) je priamo úmerná aktuálnej teplote. Ak teplota klesne na 0, prehľadávanie sa správa presne tak, ako hill climbing.

Algoritmus pracuje nasledovne. Prehľadávanie začína podobne ako u hill climbing zo stavu, ktorý môže byť generovaný náhodne (prípadne heuristicky). V každej iterácii je preskúmané náhodne vybraté susedné riešenie. Ak je toto riešenie lepšie ako aktuálne, potom sa stáva novým aktuálny stavom. Ak je horšie vzhľadom na optimalizačnú funkciu, potom je akceptované ako aktuálne riešenie s pravdepodobnosťou priamo úmernou vyššie spomínanej teplote. Ak je toto riešenie odmietnuté, potom sa preskúma iné susedné riešenie podobným spôsobom. Pri tomto postupe je pravdepodobnosť dosiahnutia lepšieho riešenia vyššia než u hill climbing.

Podobne ako u metódy hill climbing, aj u simulovaného žíhania je efektívnosť tejto metódy silne závislá od definície funkcie susednosti. Naviac veľmi dôležitú úlohu zohráva plán ochladzovania. Ak je teplota znižovaná príliš rýchlo, nemusí sa tým veľmi zvýšiť pravdepodobnosť nájdenia lepšieho riešenia. Na druhej strane čím pomalšie ochladzovanie, tým dlhší čas je potrebný na ukončenie behu programu.

6.4.6 Prehľadávanie tabu

Aj keď bol tento postup vyvinutý v rámci komunity operačného výskumu, metóda prehľadávanie tabu [Jánošíková 1994] je veľmi podobná hill climbing. Je taktiež používaná na riešenie optimalizačných úloh, u ktorých je dostatočné suboptimálne riešenie. Podobne ako simulované žíhanie, aj prehľadávanie tabu sa snaží vyviaznuť z lokálnych optím.

Metóda hill climbing končí dosiahnutím lokálneho optima, ktoré spravidla nie je globálnym. Metóda prehľadávanie tabu prekonáva toto obmedzenie a po dosiahnutí lokálneho optima pokračuje v hľadaní lepšieho riešenia. Inými slovami povolí prechod k novému riešeniu, ktoré je vzhľadom na zadanú optimalizačnú funkciu horšie, ako aktuálne. Prechodom k horšiemu riešeniu je však nutné zabrániť tomu, aby sa v nasledujúcom kroku metóda vrátila k predchádzajúcemu, lepšiemu riešeniu. Aby sa zabránilo zacykleniu metódy, teda návratu k preskúmaným riešeniam, tento zákaz sa musí vzťahovať nielen na posledný prechod (transformáciu), ale na t posledných prechodov ($t \in N$). To znamená, že z okolia aktuálneho riešenia sa vylúčia tie riešenia, ku ktorým by sa dospelo zakázanými (tabu) prechodmi. Podmienkou ukončenia algoritmu môže byť napríklad vyčerpanie všetkých možných prechodov z najlepšieho aktuálneho stavu, alebo prekročenie maximálneho povoleného počtu iterácií pre jeho aktualizáciu.

Prehľadávanie tabu je potrebné vidieť ako triedu algoritmov, ktoré sú charakteristické tým, že sa v nich určitým spôsobom definuje a spravuje zoznam tabu, ktorý obsahuje popis zakázaných prechodov. Môže to byť napr. zoznam do daného okamžiku už preskúmaných uzlov, alebo zoznam zakázaných smerov postupu a pod. Po každom prechode je upravený aj zoznam tabu. Rôzne algoritmy prehľadávania tabu môžu využívať rôzne stratégie manipulácie so zoznamom tabu.

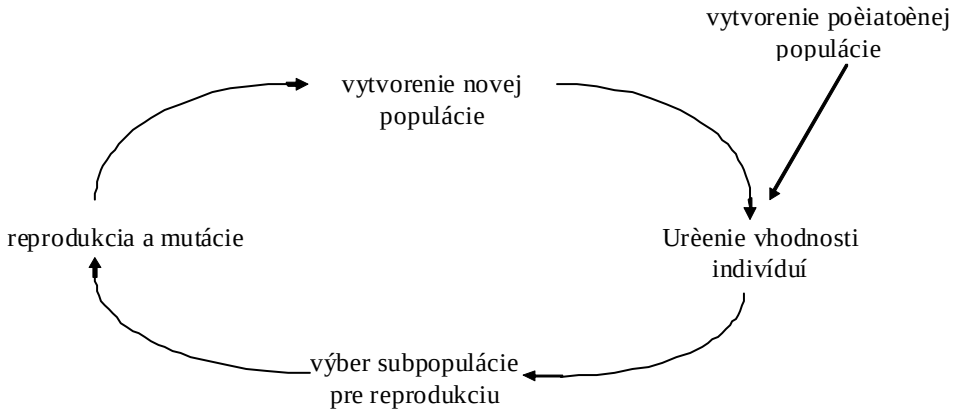
Efektívnosť prehľadávania tabu v porovnaní s hill climbing závisí len od spôsobu, akým je definovaný a spracovávaný zoznam tabu. Keďže v tomto smere nie sú žiadne obmedzenia, tabu prehľadávanie je takto veľmi všeobecnou stratégiou riešenia.

6.4.7 Genetické algoritmy

Táto metóda vychádza z Darwinovej evolučnej teórie, uvažujúc sexuálnu reprodukciu kombinovanú s náhodnou mutáciou [Mach 1996].

Potenciálne riešenie je reprezentované ako jedno indivíduum populácie. V používanej terminológii je nazývané chromozómom a jednotlivé časti (parametre) riešenia sú gény. Chromozóm kóduje riešenie špecifického problému jednoduchou štruktúrou, najčastejšie ako reťazec binárnych hodnôt. Klasická podoba algoritmu (základný cyklus je vyobrazený aj na

obrázku Obr. 31) vyzerá nasledovne.



Obr. 31 Základný cyklus genetického algoritmu

```
Genetický algoritmus:

   1. Náhodné generovanie počiatočnej populácie

   2. Určenie vhodnosti každého indivídua populácie

Opakuj
   3. Určenie pravdepodobnosti výberu každého indivídua

   4. Výber subpopulácie indivíduí pre reprodukciu

   5. Vznik nových indivíduí náhodnou reprodukciou

   6. Náhodná mutácia nových indivíduí

   7. Určenie vhodnosti nových indivíduí

   8. Vytvorenie novej aktuálnej populácie

Pokiaľ podmienka ukončenia
```

Pri výpočte pravdepodobnosti výberu indivídua sa najprv určí priemerná vhodnosť populácie a na základe tejto sa normalizuje vhodnosť každého indivídua. Proporcionálne takto vzniklej hodnote sa stanoví hľadaná pravdepodobnosť. Tým sa dosahuje, že sľubnejšie indivíduá získavajú lepšie možnosti reprodukcie.

Z vybraných indivíduí sa náhodne zvolia dvojice rodičov a ich rekombinovaním vznikajú dvojice potomkov (veľkosť populácie ostáva konštantná). Mutácii sa prikladá rádovo menší význam ako kríženiu. Používa sa však preto, že pri krížení nevzniká nový genetický materiál, iba sa distribuuje, a práve mutácia môže zaistiť jeho tvorbu. Zvyčajne je realizovaná náhodnou inverziou náhodného bitu reťazca chromozómu.

V praxi sa zaužívali dva spôsoby vytvárania novej populácie. Jeden z nich (generatívny) nahrádza všetky indivíduá starej aktuálnej generácie novými

indivíduami. Pri druhom sa nová generácia tvorí z novovzniknutých indivíduí a z indivíduí starej generácie výberom indivíduí s najvyššou vhodnosťou.

Podmienka ukončenia algoritmu je zvyčajne odvodená z priemernej vhodnosti celej populácie. Ako populácia konverguje, priemerná vhodnosť populácie sa blíži vhodnosti najlepšieho indivídua. Nie je však žiadna garancia nájdenia globálneho optima.

Aby bolo možné použiť genetický algoritmus pre riešenie úloh rozvrhovania, musí sa najprv nájsť reťazec, ktorý reprezentuje možné rozvrhy. Všeobecne používané sú binárne reťazce. Napr. pre príklad z úvodu tejto kapitoly by bolo možné reprezentovať počiatočné časy jednotlivých operácií, t.j. napr. binárny reťazec veľkosti 14 x 5 bitov pre reprezentáciu 14 operácií. Pomocou 5 bitov možno reprezentovať číslo veľkosti 0 až 31, takže asi najvýhodnejšie bude reprezentovať posunutie začiatku danej operácie oproti jej najskoršiemu možnému začiatku. Napr.: 00011 01000 01010 … 00111 možno považovať za reprezentáciu rozvrhu, v ktorom operácia T_{11} začne v čase 10=7+3(00011), T_{12}=9+8(01000), atď. Parameter zvaný vhodnosť reťazca bude u rozvrhovania daný kvalitou rozvrhu, ktorý reprezentuje (t.j. vyjadruje správnosť rozvrhu, ako aj optimalizačné kritérium). Hneď teraz je dôležité si uvedomiť, že niektoré reprezentácie môžu byť lepšie ako iné a voľba správnej reprezentácie často znamená obrovskú pomoc pri prehľadávaní.

V rozvrhovaní vygenerovaný reťazec nemusí reprezentovať prípustný rozvrh. Jedným spôsobom ako sa vyrovnať s týmto problémom je pridať penalizačnú funkciu k parametru vhodnosti, ktorá bude vyjadrovať závažnosť porušenia ohraničení pri danom rozvrhu. Iná možnosť je “opraviť” každý reťazec generovaný algoritmom (napr. pomocou hill climbing).

Vo všeobecnosti sa od genetických algoritmov očakáva, že majú väčšiu šancu preskúmať väčšiu časť priestoru prehľadávania než hill climbing. Avšak ako to vyplýva z ich podstaty, vyžadujú vopred nejasný počet iterácií aby našli riešenie dobrej kvality, takže vo všeobecnosti pre nájdenie riešenia je potrebný netriviálny čas[1].

Aj genetické algoritmy majú problém konvergencie, ktorý je do istej miery analogický problému uviaznutia v lokálnych optimách pri metóde hill climbing. Pretože stavebné bloky vhodnejších reťazcov majú väčšiu šancu, že prežijú v ďalšej generácii, je tu určité nebezpečenstvo, že všetky reťazce budú mať tie isté stavebné bloky. Preto je nutné nájsť určitú rovnováhu medzi mutáciami v algoritme a väčšími šancami pre vhodnejšie reťazce stať sa rodičmi.

U genetického algoritmu je nutné zvoliť rad parametrov ako veľkosť populácie, veľkosť množiny rodičov, počet krížencov generovaných každým rodičovským párom, atď. Rovnako je nutné zvoliť operátory, ktoré genetický algoritmus použije na výber rodičov, rekombináciu, mutácie atď. Naviac je možné zmeniť aj vyššie popísanú riadiacu stratégiu.

[1] Týmto pojmom označme také algoritmy, ktorých časová zložitosť sa nedá vyjadriť ako funkcia veľkosti vstupu.

6.4.8 Neurónové siete

Neurónové siete sa ukázali ako nástroj vhodný na riešenie úloh spĺňania ohraničení, vrátane optimalizačných úloh [Tsang 1993]. Použitím veľkého počtu jednoduchých procesorov sa dosahuje schopnosť generovať rozvrhy rýchlejšie než je to možné ktoroukoľvek z ostatných spomínaných metód. Avšak jedným dôležitým obmedzením pri tomto prístupe je, že vybudovanie špeciálnej siete pre riešenie konkrétnej aplikácie je obvykle drahé.

Pri tomto prístupe je problém reprezentovaný ako sieť. Spôsob činnosti jednotlivých uzlov v sieti, ako aj spôsob ich vzájomného prepojenia sú kľúčom k úspechu tejto metódy.

Ako príklad možno uviesť systém Genet [Tsang 1993]. Aby ho bolo možné použiť, musí sa najprv úloha formulovať ako úloha spĺňania ohraničení. V systéme Genet každá hodnota premennej je reprezentovaná ako jeden (hodnotový) uzol a každé nebinárne ohraničenie tiež ako uzol (ohraničenia). Binárne ohraničenia sú reprezentované priamou inhibítorovou väzbou medzi hodnotovými uzlami. Každé n-árne ($n > 2$) ohraničenie je reprezentované už spomínaným uzlom ohraničenia, ktorý je spojený s každým relevantným hodnotovým uzlom.

Množina pravidiel je navrhnutá tak, aby zabezpečila, že sieť sa ustáli v určitom stave. Jednoduchý mechanizmus učenia (typu reinforcement) je použitý na vyviaznutie z lokálnych optím.

Pre riešenie binárnych úloh spĺňania ohraničení (premenné s konečnými doménami a ohraničenia len unárne a binárne) je postup veľmi jednoduchý. Neurónová sieť sa inicializuje priradením váh -1 všetkým hranám (všetky uzly sú v tomto prípade hodnotové). Uzly reprezentujúce rôzne hodnoty tej istej premennej tvoria samostatnú podmnožinu (cluster). Pre každú takúto podmnožinu sa jeden náhodne vybratý uzol vybudí (t.j. priradí sa mu váha 1), všetky ostatné majú váhu 0. Spustí sa výpočet v sieti, až do ustálenia, pričom v každej podmnožine bude vybudený ten uzol, ktorý má najväčšiu hodnotu na vstupe.

Dosiaľ vykonané testy na rôznych úlohách [Tsang 1995] ukazujú, že Genet má vyššiu úspešnosť v nájdení riešenia pre testované riešiteľné úlohy než najlepšie známe algoritmy hill climbing vyvinuté pre tieto úlohy. Odhady hovoria, že systémom Genet by sa mali dať riešiť pomerne veľké úlohy rádovo v sekundách.

6.4.9 Expertné systémy

Expertné systémy majú dlhú históriu použitia pre účely rozvrhovania. Jeden z najznámejších expertných systémov pre rozvrhovanie je ISIS [Fox 1987]. Väčšina týchto systémov je pravidlovo orientovaných a ich prínosy sú spravidla špecifické pre ich doménu použitia, na ktorú sú tieto systémy priam “ušité”. Preto môžu byť expertné systémy použité v princípe na ľubovoľný typ úloh, ktoré sú riešiteľné.

Architektúra takéhoto expertného systému je obyčajne veľmi jednoduchá. V zásade ide o množinu pravidiel (báza znalostí) a inferenčný mechanizmus, ktorý riadi spôsob a poradie použitia jednotlivých pravidiel. Sila expertných systémov teda tkvie v kvalite znalostí z danej domény použitia

reprezentovaných v báze znalostí.

Zásluhou jasne formulovaných pravidiel, ktoré sú základom bázy znalostí takéhoto expertného systému, sú tieto systémy pomerne dobre zrozumiteľné pre používateľov. Tieto pravidlá sa získavajú od experta pre daný rozvrhovací problém a to je práve najkritickejšie miesto tejto metódy (získať a korektne formulovať vedomosti experta formou pravidiel je veľmi zložité).

Tiež voľba inferenčného mechanizmu a stratégie riešenia konfliktov medzi pravidlami (ak sú v danom okamžiku aplikovateľné viaceré pravidlá naraz), môže byť ďalším problémom. Štandardné inferenčné mechanizmy ponúka celá rada komerčných "shell-ov" pre návrh expertných systémov s pomocou ktorých môže byť vývoj aplikácie dosť urýchlený.

6.4.10 Systémy na programovanie ohraničení

Niektoré z metód stručne popísaných v predchádzajúcich bodoch boli zabudované do niekoľkých komerčných programovacích systémov ohraničení. Vynikajúci prehľad týchto systémov možno nájsť v [Cras 1993]. Ide vlastne o rozličné programovacie jazyky, ktoré umožňujú efektívne vyjadrenie a riešenie úloh spĺňania ohraničení. Väčšina týchto systémov sú jazyky CLP, iná skupina sú potom objektovo orientované jazyky.

Tieto systémy poskytujú účinné techniky spĺňania ohraničení bez toho, aby ich musel používateľ poznať. Aj keď na druhej strane ich znalosť môže požívateľovi pomôcť zlepšiť efektívnosť vyvíjanej aplikácie.

Napríklad CLP jazyky (*CHIP* [Beldiceau, Contejean 1994], *Prolog III* [Benhamou 1993], *ECL^{i}PS^e* [Apt, Wallace 2006] a iné) poskytujú na riešenie numerických ohraničení pre premenné s reálnymi doménami lineárne programovanie, pre premenné s konečnými doménami algoritmy spĺňania ohraničení a pre účely optimalizácie metódu vetvenia a medzí. Naviac boli vyvinuté nové, špecializované algoritmy a prístupy (napr. [Aggoun, Beldiceaunu 1993] pre *CHIP*). Podrobnejší opis niektorých prístupov zameraných na riešenie disjunktných ohraničení v jazyku *ECL^{i}PS^e* možno nájsť v [Paralič 1997].

Jazyky s objektovo orientovaným prístupom, ako napr. *ILOG solver* [Puget 1994] ponúkajú podobné techniky ako CLP, naviac sú flexibilnejšie čo sa týka riadiacich stratégií, nakoľko je možné použiť aj neúplné metódy prehľadávania, čo však na druhej strane vyžaduje experta na metódy riešenia rozvrhovacích úloh, ktorý by vedel využiť túto flexibilitu.

6.5 Optimalizácia v úlohách rozvrhovania

Dôležitým aspektom rozvrhovacích úloh je optimalizácia, t.j. hľadanie takého riešenia, ktoré nielen že spĺňa všetky technologické ohraničenia, ale je aj optimálne podľa daného kritéria (viď. časť 6.1). V prostredí logického prgramovania ohraničení (CLP) sa na tento účel prakticky výlučne používa adaptovaná verzia algoritmu metódy vetvenia a medzí zastrešená vhodným predikátom s príslušným počtom argumentov.

Metóda vetvenia a medzí používa pre orezávanie tzv**. *hornú a dolnú hranicu nákladov***, t.j. predpokladá, že optimálne riešenie leží niekde medzi

týmito hranicami. Na tomto mieste si je dôležité uvedomiť, že okrem spôsobu a efektívnosti prehľadávania samotného algoritmu vetvenia a medzí môže celkový čas potrebný na nájdenie riešenia výrazne ovplyvniť aj voľba týchto hraníc na začiatku prehľadávania. Čím užší je inicializačný interval, tým menší priestor prehľadávania musí spracovať metóda vetvenia a medzí.

Efektívna voľba týchto hraníc je silne závislá od konkrétnych znalostí o riešenej rozvrhovacej úlohe. Pre rozsiahle úlohy je obvykle nevyhnutné venovať určitý čas na výpočet čo možno najlepšieho odhadu pre dolnú a hornú hranicu. Tento čas sa obvykle mnohonásobne vráti v ďalšej etape výpočtu, keď nastúpi metóda vetvenia a medzí.

Vygenerovať počiatočné riešenie (t.j. stanoviť ***hornú hranicu riešenia***) je možné pomerne jednoducho deterministickým algoritmom [Caseau, Laburthe 1995]. Rozvrh sa vytvára postupne tak, že sa vyberajú úlohy (operácie) jedna za druhou a každá z nich začne tak skoro, ako je to len možné.

V každom kroku je k dispozícii množina úloh, z ktorých je ešte možné vybrať nasledujúcu. Na začiatku sú v tejto množine všetky úlohy (operácie). V každom kroku sa vyberie jedna z týchto úloh a priradí sa jej najskorší možný začiatok. Táto úloha sa potom vyberie zo spomínanej množiny. Celý postup sa potom opakuje tak dlho, až kým sa všetky úlohy z množiny nevyberú (viď. časť 6.2.2). Ťažisko algoritmu teda leží v pravidle, podľa ktorého sa vyberá úloha z množiny ešte nerozvrhnutých úloh. Ako príklad uveďme nasledovné heuristiky:

FIFO vyberaj zaradom podľa poradia (úlohy sú usporiadané podľa postupnosti v rámci jednotlivých výrobkov)

EST vyber úlohu (operáciu) s najskorším možným časom začiatku

LST vyber úlohu (operáciu) s najneskorším možným časom začiatku

EFT vyber úlohu (operáciu) s najskorším možným časom ukončenia

LFT vyber úlohu (operáciu) s najneskorším časom ukončenia

SPT vyber úlohu (operáciu) s najkratším trvaním

LPT vyber úlohu (operáciu) s najdlhším trvaním

MWR vyber úlohu (operáciu) s najdlhšou zvyškovou prácou (súčet trvaní úloh, ktoré ešte musia byť vykonané za vybranou úlohou)

Okrem odhadu hornej hranice, kde ide vlastne o nájdenie čo možno najlepšieho suboptimálneho riešenia, môže prispieť k zúženiu priestoru prehľadávania aj dobrý odhad ***dolnej hranice***. Tu ide o odhad doby, pod ktorú sa určite nedá už rozvrh stihnúť.

Pre odhad dolnej hranice sa používa klasický postup, ktorý spočíta dĺžky trvaní jednotlivých úloh (operácií) pre jednotlivé procesory (stroje) a pre jednotlivé zákazky. Za dolnú hranicu sa potom vyberie najdlhší z nich. Znamená to, že rozvrh nemôže byť kratší ako súčet trvaní všetkých úloh na tom procesore, ktorý bude najviac využívaný, resp. ako súčet trvaní všetkých úloh tej zákazky, ktorá ho má najväčší. Túto hodnotu je možné ešte zvýšiť o najskorší možný čas začiatku spomedzi všetkých úloh (operácií) na tomto najkritickejšom procesore, resp. na tejto zákazke (ten sa už mohol

v priebehu prvotnej propagácie ohraničení zvýšiť).

Dôvod, prečo sa v CLP jazykoch používa práve tento prístup, je jednoduchý. Metóda vetvenia a medzí je úplná metóda prehľadávania a výborne sa hodí do prostredia kde sú možné návraty (z anglického backtracking), ktoré sú typickým sprievodným javom prehľadávania do hĺbky u logických programovacích jazykov. V princípe existujú dve stratégie (sú implementované napr. V CLP jazykoch *CHIP* a *ECLiPSe*):

MINMAX Vychádzajúc zo známej počiatočnej hodnoty hornej a dolnej hranice nákladov, ktoré sa požadujú od riešenia $C^{\max}$ a $C^{\min}$ používa ohraničenie $C^{\min} \le C \le C^{\max}$ (ide o celé čísla) na orezávanie priestoru prehľadávania. Akonáhle nájde riešenie s nákladmi C_n, prehľadávanie zastaví a začne opäť odznova, ale s novým ohraničením $C^{\min} \le C \le C_n - 1$. Ak sa nenájde žiadne ďalšie riešenie, alebo ak $C_n = C^{\min}$, potom posledne nájdené riešenie je optimálne.

MINIMIZE pracuje veľmi podobne s tým rozdielom, že po nájdení riešenia nezačína prehľadávať od začiatku, ale pokračuje na mieste, kde sa práve nachádza.

Na jednej strane MINIMIZE v porovnaní s predchádzajúcim MINMAX nemusí odznova prehľadávať tú časť priestoru, ktorú už predtým prehľadal do nájdenia posledného riešenia. Na druhej strane sa pri tejto metóde objavuje u mnohých úloh jav v angličtine nazývaný trashing v prípade, že množstvo riešení s podobnými nákladmi je topologicky blízko seba v priestore prehľadávania.

Tento postup je možné vylepšiť dvoma spôsobmi.

1. Ak sa uspokojíme so suboptimálnym riešením v rámci vopred zadanej presnosti *E* (číslo od 0 do 1), potom je možné oba vyššie uvedené postupy upraviť nasledovne. Po nájdení nového riešenia C_n sa novou hornou hranicou stane $C_n(1-E)-1$ (namiesto pôvodnej $C_n - 1$). Ak sa nenájde žiadne lepšie riešenie, potom vieme, že optimálne riešenie leží v intervale $\langle C_n(1-E), C_n \rangle$. Tento prístup čiastočne predchádza javu nazvanému vyššie trashing.
2. Paralelným prehľadávaním priestoru prehľadávania.

6.6 Porovnanie metód na riešenie úloh rozvrhovania

6.6.1 Rozdelenie metód do skupín podľa príbuznosti

Z analýzy popísaných metód riešenia úloh rozvrhovania vyplýva, že v zásade môžeme tieto metódy rozdeliť na tri skupiny.

1. ***Úplné metódy***, ktoré zaručujú nájdenie (optimálneho) riešenia ak existuje (lineárne programovanie, metóda vetvenia a medzí, spĺňanie ohraničení).
2. ***Neúplné metódy*** (hill climbing, simulované žíhanie, tabu search, genetické algoritmy), ktoré neprehľadávajú celý priestor prehľadávania, iba jeho časť s tým že zaručujú iba nájdenie suboptimálneho riešenia.

3. Iné, ***neštandardné metódy***, kam možno zahrnúť neurónové siete a expertné systémy.

Aj keď je druhá skupina metód často veľmi účinná pre rozsiahle úlohy, kde metódy z prvej skupiny narazia na problém kombinatorickej explózie, je potrebné si uvedomiť, že stochastické (neúplné) metódy fungujú dobre len pre úlohy, kde podobné (susedné) riešenia majú aj približne rovnaké náklady. Ak totiž nie je vzťah medzi podobnými riešeniami a ich nákladmi, potom niet dôvodu, prečo by mali byť stochastické metódy lepšie než štandardný backtracking (t.j. najjednoduchší algoritmus pre úplné prehľadávanie).

Pojem "podobnosti" medzi riešeniami je zachytený v operátoroch definovaných pre danú úlohu. U hill climbing a simulovaného žíhania ide o funkcie susednosti, ktoré generujú nové (susedné) riešenia, u genetických algoritmov zase operátor rekombinácie, ktorý generuje nové riešenie ako vhodnú kombináciu dvoch už nájdených riešení.

6.6.2 Kritériá pre výber najvhodnejšej metódy

Aby sa riešiteľ rozvrhovacej aplikácie mohol rozhodnúť, ktorú metódu použiť, potrebuje do hĺbky poznať tak riešený problém, ako aj jednotlivé metódy. Pri tejto analýze je nutné zodpovedať niektoré základné otázky a z odpovedí na ne by mali vyplynúť základné doporučenia z hľadiska použiteľnosti jednotlivých metód. Existujú štyri základné kritériá.

1. Splniteľnosť alebo optimalizácia

Najprv je dôležité si uvedomiť, či je cieľom nájsť akékoľvek riešenie spĺňajúce všetky zadané ohraničenia (úlohy splniteľnosti), alebo je potrebné nájsť riešenie, ktoré je optimálne z hľadiska nejakého kritéria (optimalizačné úlohy).

Lineárne programovanie (len pre úlohy, kde sú ohraničenia aj kriteriálna funkcia lineárne), metóda vetvenia a medzí, hill climbing, simulované žíhanie, tabu search a genetický algoritmus sú určené najmä pre riešenie optimalizačných úloh. Spĺňanie ohraničení na úlohy splniteľnosti. Expertné systémy a neurónové siete sa dajú vybudovať na dosiahnutie akéhokoľvek potrebného cieľa.

Tu je nutné zmieniť sa ešte o jednom dôležitom a častom fenoméne, a síce preferenčných ohraničeniach. Často totiž v rozvrhovaní sú definované ohraničenia, ktoré musia byť splnené (***tvrdé, alebo technologické ohraničenia***), ale aj ohraničenia, ktorých splnenie by bolo síce vítané, ale nie je nevyhnutné (***mäkké, alebo preferenčné ohraničenia***).

V takomto prípade sú možné v zásade dva prístupy:

- považovať preferenčné ohraničenia za tvrdé a problém sa stáva problémom splniteľnosti. Ak riešenie neexistuje, potom je možné vybrať niektoré preferenčné ohraničenia a uvolniť ich (nebrať do úvahy),
- porušenie preferenčného ohraničenia zarátať do nákladov a celý problém riešiť ako optimalizačný s tým, že sa minimalizujú náklady (a teda počet porušených preferenčných ohraničení).

2. Výpočtový čas verzus optimálnosť riešenia

Pre nájdenie zaručene optimálneho riešenia je nutné použiť niektorú z úplných metód. Avšak tie narážajú na kombinatorickú explóziu. Čas výpočtu totiž exponenciálne narastá s veľkosťou úlohy (veľkosť úlohy je daná počtom množín disjunktných operácií a ich veľkosťou).

Stochastické metódy si poradia aj s rozsiahlejšími úlohami, ale zaručujú len suboptimálne riešenie. Preto je nutné zvážiť, ktorá požiadavka je dôležitejšia. Napríklad pri dlhodobom plánovaní ktoré narába s drahými zdrojmi, nie je až taký rozhodujúci čas výpočtu, ale práve optimálnosť riešenia. V takom prípade je potrebné použiť niektorú z úplných metód.

Naopak sú zasa situácie (napr. v mimoriadnych stavoch), kedy je úplne postačujúce suboptimálne riešenie, ktoré je ale možné získať čo najrýchlejšie. V takom prípade má najväčšie šance neurónová sieť nasledovaná metódou hill climbing. Simulované žíhanie a tabu search budú asi potrebovať viac času, čo je cena za nájdenie lepšieho riešenia.

3. Špecifikácia problému

Každá z vyššie popísaných metód si vyžaduje svoju reprezentáciu úlohy. Lineárne programovanie narába s množinou lineárnych nerovníc a kriteriálnou funkciou.

Metódu vetvenia a medzí možno použiť na každý optimalizačný problém, ak je k dispozícii spôsob, ako ohodnotiť kvalitu čiastočného riešenia. To je málokedy problémom, ale efektívnosť prehľadávania závisí od presnosti tohoto ohodnotenia.

Spĺňanie ohraničení sa používa najmä pre úlohy s konečnými doménami, aj keď niektoré z nich sú aplikovateľné aj na reálne domény.

Hill climbing, simulované žíhanie a prehľadávanie tabu možno aplikovať na široké spektrum úloh. Ich kvalita záleží od funkcií susednosti, ktoré systém používa pre nájdenie nového riešenia (u simulovaného žíhania je naviac veľmi dôležitý plán ochladzovania a u prehľadávania tabu spôsob vytvárania a narábania s tabu zoznamom).

Efektívnosť genetických algoritmov veľmi záleží na tom, ako sú reprezentovaní kandidáti na riešenie a ako je definovaná vyhodnocovacia funkcia, čo si vyžaduje expertízu riešiteľa rozvrhovacej aplikácie.

Bolo ukázané, že neurónové siete sú použiteľné na riešenie úloh ktoré možno formulovať ako konečné úlohy spĺňania ohraničení, a to tak bez, ako aj s požiadavkou na optimalizáciu.

4. Voľba algoritmu a implementácia

Algoritmy lineárneho programovania, vetvenia a medzí a spĺňania ohraničení sú dobre definované v literatúre a ich implementácia je pomerne priamočiara, aj keď nie vždy jednoduchá. U spĺňania ohraničení je naviac potrebné vybrať si z veľkého množstva existujúcich algoritmov (v [Mach, Paralič 2000] možno nájsť pomerne rozsiahly prehľad existujúcich algoritmov a početné odkazy na literatúru), čo si vyžaduje značné vedomosti.

Podobne aj základný algoritmus pre hill climbing je dobre definovaný, ale efektívnosť tejto metódy je silne závislá od kvality funkcie susednosti. Jej definícia leží na riešiteľovi danej rozvrhovacej úlohy (nájsť nejakú nie je obvykle až taký problém, ale nájsť takú, ktorá bude efektívne prehľadávať priestor, je ťažká úloha).

Simulované žíhanie naviac oproti hill climbing vyžaduje aj definíciu plánu ochladzovania, ktorý je kritickou zložkou algoritmu. Náročnosť implementácie týchto algoritmov je daná hlavne zložitosťou použitej funkcie susednosti. U prehľadávania tabu ešte naviac aj zložitosťou tabu zoznamu a jeho manipulačného mechanizmu.

Expertné systémy môžu byť vytvorené s použitím existujúcich shell-ov relatívne ľahko, ale nájdenie efektívnych pravidiel je obvykle veľmi zložité.

Veľká výhoda systémov na programovanie ohraničení je v tom, že riešiteľ rozvrhovacej aplikácie môže využívať metódy lineárneho programovania, metódu vetvenia a medzí, algoritmy spĺňania ohraničení a prípadne ďalšie metódy bez toho, aby sa ich musel učiť a sám implemetoval.

Členenie metód vzhľadom na uvedené kritériá je zhrnuté v Tab. 10.

Zaujímavými sa z pohľadu tohto zhrnutia a porovnania metód na riešenie úloh rozvrhovania javí logické programovanie ohraničení (CLP), ktoré v sebe integruje množinu metód z prvej skupiny.

CLP totiž v sebe integruje algoritmy lineárneho programovania (pre reálne prípadne racionálne premenné), metódu vetvenia a medzí (hľadanie optimálneho riešenia) a spĺňanie ohraničení (pre celočíselné a enumeračné premenné). Naviac je omnoho pružnejšie v reprezentácii netypických ohraničení, s ktorými sa často stretávame v konkrétnych aplikáciách. V prostredí CLP je možné implementovať veľmi prirodzene aj expertné systémy.

Ako už bolo spomínané, vďaka svojej deklaratívnosti tak ponúka CLP veľmi účinný prostriedok pre riešenie úloh rozvrhovania, ako aj experimentovanie s rôznymi postupmi bez toho, aby bolo nutné programovať špeciálne algoritmy. A to všetko pri podstatne menšom rozsahu zdrojového kódu, než je tomu napr. u tradičných procedurálnych programovacích jazykov.

Metódy a systémy založené na spĺňaní ohraničení umožňujú návrh presných, flexibilných, efektívnych a rozšíriteľných rozvrhovacích aplikácií.

Presné a flexibilné sú tieto aplikácie preto, že môžu brať do úvahy každé ohraničenie, ktoré je možné vyjadriť v danom jazyku spĺňania ohraničení. Zároveň systémy CLP ponúkajú stále širšiu množinu ohraničení (nielen numerické, ale aj rôzne symbolické ohraničenia), ako aj nástroje na definovanie nových ohraničení, podľa potrieb používateľa.

Efektívne sú tieto aplikácie do tej miery, nakoľko efektívne algoritmy šírenia a spĺňania ohraničení sú k dispozícii v tom ktorom systéme.

Rozšíriteľnosť týchto aplikácií je daná tým, že požiadavka na nový typ ohraničenia obvykle vedie iba k definícii tohto nového ohraničenia a prípadne spôsobu jeho propagácie, pričom nie je nutné meniť existujúci program.

metóda	***všeobecné zásady***	***zásady špecifické pre danú metódu***
Lineárne programovanie	• *splniteľnosť aj optimalizácia* • *úplné*	• *problém musí byť zadaný formou množiny rovníc a nerovníc*
Metóda vetvenia a medzí	• *optimalizácia* • *úplné*	• *vyžaduje heuristiky na orezávanie* • *dôležité je poradie prehľadávania vetiev*
Spĺňanie ohraničení	• *splniteľnosť* • *úplné aj neúplné*	• *existuje veľké množstvo algoritmov* • *vhodné najmä pre netriviálne ohraničenia*
hill climbing	• *splniteľnosť a optimalizácia, ak stačí suboptimum* • *hill climbing môže uviaznuť v lokálnych optimách* • *simulované žíhanie a tabu search sa z nich snažia vyviaznuť*	• *vyžaduje funkciu susednosti, ktorá je rozhodujúca pre efektívnosť*
Simulované žíhanie		• *funkcia susednosti rozhoduje o efektívnosti* • *plán ochladzovania je kritický*
tabu search		• *efektívnosť závisí najmä od stratégie manipulácie s tabu zoznamom*
Genetické algoritmy	• *optimalizácia* • *neúplné*	• *rozhodujúca je reprezentácia* • *efektívnosť môže byť citlivá na voľbu hodnôt parametrov a operátorov*
Neurónové siete	• *splniteľnosť alebo optimalizácia* • *rýchly výpočet*	• *zostavenie siete a mechanizmus zmeny hodnôt sú rozhodujúce* • *vytvorenie špeciálnej siete môže byť veľmi drahé*
Expertné systémy	• *šité na mieru* • *sila je v doménovo závislých znalostiach*	• *nutné je získanie vedomostí od experta a to môže byť zložité*

Tab. 10 Zásady, ktoré je potrebné brať do úvahy pri výbere metódy na riešenie úloh rozvrhovania

Ďalšími dôležitými parametrami, ktoré je nutné brať do úvahy pri riešení úloh rozvrhovania, sú celkový časový interval, pre ktorý sa robí rozvrh a časová presnosť rozvrhu (elementárny časový interval, ktorý sa pri rozvrhu uvažuje). Ak ich vzájomný pomer je malý (napr. jeden deň v hodinách), je vhodnejšie voliť diskrétnu reprezentáciu času, ktorá dovoľuje priamejší prístup k reprezentovaným údajom. Na druhej strane, ak je tento pomer veľký (napr. jeden mesiac v sekundách), najlepšie je voliť reprezentáciu so spojitou reprezentáciou času.

Sila technológie CLP je v aktívnej úlohe definovaných ohraničení, ktoré zužujú domény premenných na ktoré sú aplikované a tým aj priestor, ktorý v ďalšom bude potrebné prehľadať. Pritom je dôležité si uvedomiť, že takéto pružné prostredie pre vyjadrenie rôznych druhov ohraničení dáva výborný priestor pre prototypovanie a to aj v prípade, keď výsledný systém bude naprogramovaný v inom programovacom jazyku. V tom je výhoda

v porovnaní s jednoúčelovými (síce výkonnými ale málo flexibilnými) programovacími prostriedkami.

7 Zásobovanie

Cieľom zásobovania je zistiť optimálne veľkosti zásob a spôsob optimálneho riadenia ich úrovne (pohybu). Optimalizačným kritériom je minimalizácia celkových nákladov. Pritom náklady sú v zásade dvoch typov:

1) Náklady rastúce s veľkosťou zásob, napr. náklady na udržiavanie zásob (skladovacie náklady) – n_1
2) Náklady klesajúce s veľkosťou zásob, napr. náklady vyplývajúce z nedostatku pohotových zásob – n_2

Zásoba je ľubovoľný pohotový ekonomický zdroj, ktorý nie je v danom časovom intervale plne využívaný, avšak jeho výška je stanovená tak, aby z ekonomického hľadiska umožňoval čo najvýhodnejšie krytie budúceho dopytu. Dopyt môže byť dvojaký:

1) Deterministický (vopred známy)
2) Stochastický, ktorý môže byť
   - so známym zákonom rozdelenia pravdepodobnosti, alebo
   - s neznámym zákonom rozdelenia pravdepodobnosti.

Modely zásobovania (modely riadenia zásob, viď. napr. [Dudorkin 1997]) rozdeľujeme na základe viacerých kritérií.

1. Podľa počtu rozhodnutí o dopĺňaní stavu zásob za dané obdobie:

   A) Statické (za celé obdobie sa zásoby dopĺňajú iba raz)

   B) Dynamické (za celé obdobie sa zásoby môžu dopĺňať viackrát)

2. Podľa priebehu dopytu:

   A) Deterministické (dopyt počas daného obdobia je presne známy)

   B) Stochastické (dopyt počas daného obdobia je stochastický a riadi sa teda nejakým zákonom rozdelenia pravdepodobnosti)

3. Podľa charakteru dopytu:

   A) So spojitým dopytom (dopyt sa mení spojito počas celého sledovaného obdobia)

   B) S nespojitým dopytom (počas sledovaného obdobia sa dopyt mení diskrétne, iba v určitých časových okamihoch)

4. Podľa počtu skladových položiek:

   A) Jednopoložkové (uvažuje iba jednu skladovú položku)

   B) Viacpoložkové (uvažuje viacero skladových položiek naraz)

5. Podľa počtu skladov:

   A) S jedným skladom

   B) S viacerými skladmi

Niekedy hovoríme o ***stratégii riadenia zásob***. Stratégia je jednoznačne určená dvoma parametrami (P_1, P_2). Parameter P_1 hovorí o tom, kedy objednávame (dopĺňame) zásoby:

A) $P_1 = s$, ak sa zásoby dopĺňajú v okamžiku, keď ich stav klesne pod hranicu s
B) $P_1 = t$, ak sa zásoby dopĺňajú v pravidelných časových intervaloch dĺžky t

Parameter P_2 hovorí o tom, na akú úroveň sa zásoby dopĺňajú:

A) $P_2 = S$, ak sa zásoby dopĺňajú na úroveň S
B) $P_2 = x$, ak sa zásoby dopĺňajú o množstvo x

Takže modely môžu byť týchto 4 typov: (t, S), (s, S), (t, x), alebo (s, x). V ďalšom predstavím 4 základné modely riadenia zásob podľa [Dudorkin 1997].

7.1 Model M1

Ide o stratégiu riadenia zásob typu (t, x). Je to model statický, stochastický, s nespojitým (diskrétnym) dopytom. Znamená to, že dopyt je diskrétna náhodná veličina Y so známym zákonom rozdelenia pravdepodobnosti $P(Y=y) = p(y)$. Pritom sú známe merné náklady plynúce z nedostatku jednotky pohotovej zásoby (n_2) a tiež merné náklady na jednotku nadbytočných zásob (n_1). Úlohou je určiť takú hodnotu objednaného množstva zásob x, aby celkové náklady $N(x)$ boli minimálne.

Náklady na skladovanie budú dané súčtom nákladov vyplývajúcich zo všetkých možných hodnôt dopytu, násobených pravdepodobnosťou, že práve daná hodnota dopytu nastane. Pričom môžu vzniknúť tri kvalitatívne odlišné situácie toho, v akom vzťahu budú objednané a doplnené zásoby o hodnote x k skutočnému dopytu y:

- Ak $x = y$, potom nevzniknú žiadne náklady, t.j. $N(x) = 0$, nakoľko zásoby presne pokryjú celý dopyt.
- Ak $x > y$, to znamená, že zásob bude viac ako skutočného dopytu a teda vzniknú náklady z nadbytočnch zásob, ktoré možno vypočítať nasledovne: $N(x) = n_1 . (x-y)$.
- Ak $x < y$, to znamená že zásob nebude dostatok na krytie skutočného dopytu, takže vzniknú náklady z nedostatku pohotovej zásoby, ktoré možno vypočítať nasledovne: $N(x) = n_2 . (y-x)$.

Celkové náklady teda budú nasledovné:

$$N(x) = \sum_{y=0}^{x-1} n_1 \cdot (x-y) \cdot p(y) + 0 + \sum_{y=x+1}^{\infty} n_2 \cdot (y-x) \cdot p(y)$$

Optimálnu hodnotu objednávaného množstva zásob x_o získame postupným vyčíslením nákladov $N(x)$ pre x = 0, 1, 2, … Za predpokladu, že $N(x)$ má len jedno lokálne minimum, možno ho určiť analyticky z dvojice nerovníc:

$$N(x_O) \le N(x_O - 1) \wedge N(x_O) \le N(x_O + 1)$$

Ak označíme hodnotu distribučnej funkcie rozdelenia dopytu $P(y)$, platí:

$P(y) = P(Y \le y) = \sum_{z=0}^{y} p(z)$ pre y = 0,1,... dostávame pre optimálne riešenie x_o:

$$P(x_O - 1) \leq \frac{n_2}{n_1 + n_2} \leq P(x_O)$$

To znamená, že stačí vyčísliť kumulatívne pravdepodobnosti $P(y)$, hodnotu výrazu $\frac{n_2}{n_1 + n_2}$ a zistiť, pre ktoré $x = x_O$ táto nerovnosť platí.

Príklad

V podniku má byť inštalovaný nový stroj a treba určiť, koľko má byť pri nákupe zakúpených nových náhradných dielov. Sú známe pravdepodobnosti $p(y)$ počtu výmen danej súčiastky stroja (viď. Tab. 11). Náklady na skladovanie náhradných súčiastok sú zanedbateľné. Cena 1ks náhradnej súčiastky nezáleží na objednanom množstve a činí 5 000 PJ/ks. Ale ak náhradná súčiastka nebude k dispozícii, vzniknú podniku straty 100 000 PJ/ks.

y	0	1	2	3	4	5	6	7
$p(y)$	0,9	0,05	0,02	0,01	0,01	0,01	0	0

Tab. 11 Zadané pravdepodobnosti počtu výmen súčiastky stroja.

Riešenie

Zo zadania úlohy vyplýva, že n_1 = 5 000 PJ/ks a n_2 = 100 000 PJ/ks. Takže hodnota výrazu $\frac{n_2}{n_1 + n_2} = \frac{100000}{105000} = 0{,}9524$

Okrem toho je potrebné vyčísliť kumulatívne pravdepodobnosti počtu výmen súčiastky stroja. To je urobené v Tab. 12.

y	0	*1*	***2***	3	4	5	6	7
$p(y)$	0,9	0,05	0,02	0,01	0,01	0,01	0	0
$P(y)$	0,9	*0,95*	***0,97***	0,98	0,99	1	1	1

Tab. 12 Výpočet kumulatívnych pravdepodobností počtu výmen súčiastky stroja.

Hodnota vyššie uvedeného výrazu 0,9524 sa nachádza medzi kumulatívnymi pravdepodobnosťami zodpovedajúcimi y = 1 a y = 2 výmeny náhradných súčiastok, preto x_O = 2, a teda podnik objedná 2 ks náhradných súčiastok na sklad.

7.2 Model M2

Opäť ide o stratégiu riadenia zásob typu (t, x). Je to model statický, stochastický, ale na rozdiel od M1 so spojitým dopytom. Znamená to, že dopyt y aj zásoba x sú spojité. Dopyt je teda spojitá náhodná veličina

popísaná hustotou pravdepodobnosti *f(y)* a zodpovedajúcou distribučnou funkciou *F(y)*, pričom platí:

$$F(x) = \int_0^x f(y)dy \quad \text{a} \quad \int_0^\infty f(y)dy = 1$$

Funkcia celkových nákladov *N(x)* sa v tomto prípade vypočíta nasledovne (analogicky k modelu M1, ale nakoľko ide o spojitý dopyt, sumy nahradia integrály):

$$N(x) = \int_0^x n_1(x-y)\,f(y)dy + \int_x^\infty n_2(y-x)\,f(y)dy$$

$$\frac{dN(x)}{dx} = n_1\int_0^x f(y)dy - n_2\int_x^\infty f(y)dy$$

Ak chceme nájsť extrém takejto funkcie, potom deriváciu položíme rovnú nule, t.j.:

$$\frac{dN(x)}{dx} = n_1 F(x_O) - n_2(1-F(x_O)) \overset{!}{=} 0$$

$$(n_1+n_2)\cdot F(x_O) = n_2$$

$$F(x_O) = \frac{n_2}{n_1+n_2}$$

Hodnota kľúčového výrazu teda zostáva nezmenená, ale nakoľko dopyt i zásoba sú spojiteé veličiny, môžeme v tomto prípade vyčísliť presne optimálnu hodnotu zásoby pre daný pomer nákladov typu n_1 a n_2.

Príklad

Dopyt je popísaný exponenciálnym rozdelením s hustotou pravdepodobnosti $f(y) = 0{,}2e^{-0{,}2y}$ $(y \geq 0)$. Jednotkové náklady z nedostatku pohotovej zásoby sú 100 PJ/kg a jednotkové náklady z nadbytočných zásob sú 40 PJ/kg.

Riešenie

Zo zadania úlohy vyplýva, že n_1 = 40 PJ/kg a n_2 = 100 PJ/kg. Takže dosadením do vzťahu pre výpočet optimálnej veľkosti zásob a jeho postupným vyčíslením dostávame:

$$F(x_0) = \frac{n_2}{n_1+n_2}$$

$$\int_0^{x_0} 0{,}2e^{-0{,}2y}dy = \frac{100}{140}$$

$$\left[-e^{-0{,}2y}\right]_0^{x_0} = 0{,}714$$

$$1 - e^{-0{,}2x_O} = 0{,}714$$

$$e^{-0,2x_O} = 0,286$$

$$-0,2x_O = \ln(0,286)$$

$$x_O = -5 \cdot \ln(0,286)$$

$$x_O = -5 \cdot \ln(0,286)$$

$$x_O = 6,264$$

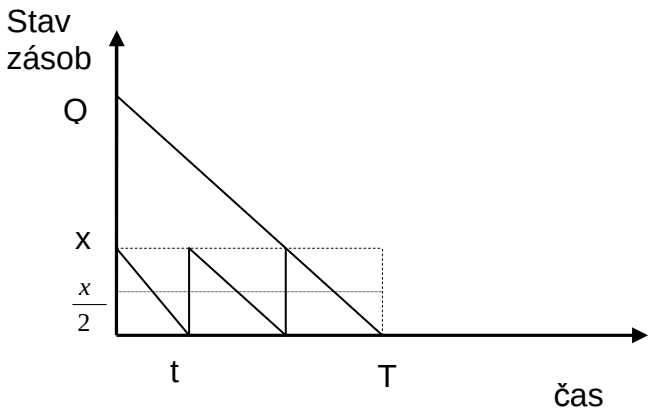
Optimálne množstvo zásob, ktoré je potrebné objednať je teda 6,264 kg.

7.3 Model M3

V tomto prípade ide o stratégiu riadenia zásob typu (s, x). Je to model dynamický, deterministický, so spojitým dopytom.

Počas dostatočne dlhého obdobia T je dopyt Q jednotiek za celé obdobie, tj. v intervale <0, T > rovnomerný a spojitý. Zásoba sa dopĺňa objednávkami rovnakej veľkosti x, a to vždy v okamžiku vyčerpania zásob (tj. s = 0). S každou jednotlivou objednávkou a dodávkou sú spojené náklady N_D, nezávisle od veľkosti objednávky. Náklady na skladovanie jednotkového množstva zásob sú n_1.

Priebeh stavu zásob je znázornený na nasledujúcom Obr. 32.



Obr. 32 Priebeh stavu zásob v prípade modelu M3

Celkové náklady $N(x)$ sú tvorené nákladmi na objednávky a nákladmi na skladovanie. Náklady na skladovanie zodpovedajú nákladom na skladovanie množstva zásob v hodnote $\frac{x}{2}$ počas celého sledovaného obdobia T (pri jednotkových nákladoch n_1). Náklady na objednávky sú dané súčinom nákladov na jednu objednávku (N_D) a počtu objednávok (ktorých musí byť $\frac{Q}{x}$, keďže dopyt za celé obdobie je Q a na jednu objednávku sa z toho získa zásoba v množstve x). Teda celkové náklady budú:

$$N(x) = N_D \frac{Q}{x} + n_1 \frac{x}{2} T$$

Pre výpočet optimálnej hodnoty x_O musíme nájsť extrém funkcie nákladov, t.j.:

$$\frac{dN(x)}{dx} = -N_D \frac{Q}{x^2} + n_1 \frac{1}{2} T = 0$$

Z čoho možno odvodiť optimálnu veľkosť objednávky x_O:

$$\frac{N_D Q}{x_0^2} = \frac{n_1 T}{2}$$

$$x_0 = \sqrt{\frac{2N_D Q}{n_1 T}}$$

Príklad

Obchodná organizácia má zabezpečiť dodávku určitého výrobku v množstve 2000 MJ/rok (MJ v tomto prípade znamená skratku pre „mernú jednotku“). Náklady na objednávku sú 250 PJ/obj. bez ohľadu na jej veľkosť. Náklady na skladovanie sú 1 PJ/MJ za rok. Určte optimálnu dodávku zásob na časový interval 1 rok.

Riešenie

Zo zadania úlohy vyplýva, že Q = 2000 MJ/rok, N_D = 250 PJ/obj., n_1 = 1 PJ/MJ za rok. Takže dosadením do vzťahu pre výpočet optimálnej veľkosti objednávaných zásob a jeho postupným vyčíslením dostávame:

$$x_0 = \sqrt{\frac{2.250.2000}{1.1}} = \sqrt{1000000} = 1000$$

Čomu zodpovedajú dve objednávky za rok, každá na 1000 MJ daného výrobku.

7.4 Model M4

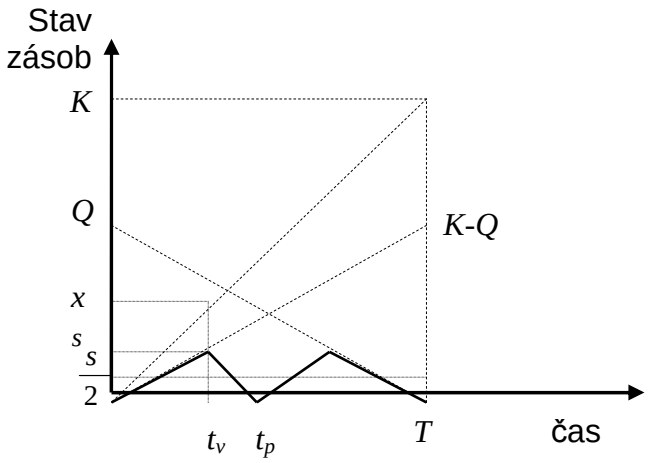
V tomto prípade ide opäť o stratégiu riadenia zásob typu (s, x). Je to model dynamický, deterministický, so spojitým dopytom, ale na rozdiel od M3 sa v tomto prípade zásoby nedopĺňajú objednávkou od externého dodávateľa, ale vlastnou výrobou.

Počas dostatočne dlhého obdobia T je dopyt Q jednotiek za celé obdobie, tj. v intervale <0, T > rovnomerný a spojitý. Nech výroba trvá stále t_v časových jednotiek a následne t_p časových jednotiek je doba potrebná k vyprázdneniu skladu. Kapacita výroby (ak by sa vyrábalo počas celého sledovaného obdobia) je $K > Q$ za sledované obdobie T. N_D sú náklady spojené so spustením výrobnej linky. Náklady na skladovanie jednotkového množstva zásob sú n_1.

Priebeh stavu zásob pre model M4 je znázornený na Obr. 33.

Celkové náklady $N(x)$ sú analogicky ako v prípade M3 tvorené nákladmi na skladovanie a nákladmi na spustenie výroby. Náklady na skladovanie zodpovedajú nákladom na skladovanie množstva zásob tentokrát v hodnote

polovica s (viď. Obr. 33) počas celého sledovaného obdobia T (pri jednotkových nákladoch n_1). Náklady na výrobu sú dané súčinom nákladov na jedno spustenie výroby (N_D) a počtu spustení výroby (ktorých musí byť $\frac{Q}{x}$, keďže dopyt za celé obdobie je Q a na jedno spustenie výroby sa získa zásoba v množstve x).



Obr. 33 Priebeh stavu zásob v prípade modelu M4

Teda celkové náklady budú:

$$N(x) = N_D \frac{Q}{x} + n_1 \frac{s}{2} T$$

Premennú s ale musíme vyjadriť pomocou jedinej riaditeľnej premennej x. To je možné urobiť z podobnosti trojuholníkov nasledovne:

$$\frac{s}{t_v} = \frac{K-Q}{T} \Rightarrow s = t_v \frac{K-Q}{T}$$

Hodnotu t_v tiež môžeme vyjadriť ako funkciu x pomocou trojuhoníkovej nerovnosti nasledovne:

$$\frac{x}{t_v} = \frac{K}{T} \Rightarrow t_v = \frac{x \cdot T}{K}$$

Spätným dosadením získame premennú s vyjadrenú len ako funkciu x:

$$s = \frac{x \cdot T}{K} \cdot \frac{K-Q}{T} = x \cdot \frac{K-Q}{K}$$

Po dosadení do funkcie celkových nákladov $N(x)$, jej zderivovaní a položení rovnej nule dostaneme hodnotu pre optimálnu veľkosť vyrobených zásob x_O:

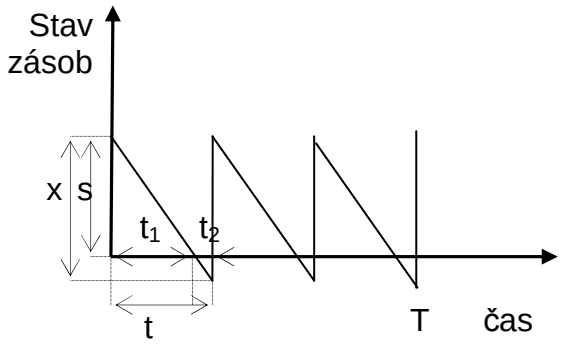
$$x_O = \sqrt{\frac{2 \cdot N_D \cdot Q \cdot K}{n_1 \cdot T(K-Q)}}$$

7.5 Model M5

Aj tento model je odvodený od modelu M3, t.j. ide opäť o stratégiu riadenia zásob typu (s, x). Je to tiež model dynamický, deterministický, so spojitým dopytom, ale na rozdiel od M3 sa v tomto prípade pripúšťa nedostatok pohotovej zásoby. Pritom ale celkový dopyt Q za sledované obdobie T musí byť uspokojený.

N_D sú náklady spojené s jednou objednávkou zásob bez ohľadu na jej veľkosť. Náklady na skladovanie jednotkového množstva zásob sú n_1. a náklady vyplývajúce z nedostatku jednotkového množstva pohotových zásob sú n_2.

Priebeh stavu zásob pre model M5 je znázornený na Obr. 34.



Obr. 34 Priebeh stavu zásob v prípade modelu M5

Výpočet celkových nákladov bude analogický ako v prípade modelu M3, t.j. náklady na objednávky (jedna objednávka stojí N_D), ktorých bude $\frac{Q}{x}$ plus náklady na skladovanie (veľkosti n_1), pri priemernom stave zásob $\frac{s}{2}$ ale len počas doby t_1 krát počet objednávkových cyklov. Oproti modelu M3 však pribudnú aj náklady z nedostatku pohotovej zásoby (veľkosti n_2), pri priemernom nedostatku zásob v hodnote $\frac{x - s}{2}$ počas doby t_2 krát počet objednávkových cyklov. Takže:

$$N(s,x) = N_D \frac{Q}{x} + n_1 \frac{s}{2} t_1 \frac{Q}{x} + n_2 \frac{(x-s)}{2} t_2 \frac{Q}{x}$$

Vo vyjadrení tejto funkcie ale musíme ešte premenné t_v a t_p prepísať pomocou riaditeľných premenných s a x. K tomu nám opäť pomôže podobnosť trojuholníkov:

$$\frac{t_1}{s} = \frac{t}{x} \Rightarrow t_1 = t\frac{s}{x}$$

$$\frac{t_2}{x-s} = \frac{t}{x} \Rightarrow t_2 = t\frac{(x-s)}{x}$$

$$\frac{T}{t} = \frac{Q}{x} \Rightarrow t = T\frac{x}{Q}$$

Takže po dosadení do funkcie celkových nákladov dostaneme:

$$N(s,x) = N_D \cdot \frac{Q}{x} + n_1 \frac{T}{2} \cdot \frac{s^2}{x} + \frac{n_2 \cdot T}{2} \cdot \frac{(x-s)^2}{x}$$

Klsickým postupom pre výpočet extrému funkcie zistíme optimálne hodnoty x_O a s_O:

$$x_0 = \sqrt{\frac{n_1 + n_2}{n_2}\frac{2N_D Q}{n_1 T}}$$

$$s_0 = \sqrt{\frac{n_2}{n_1 + n_2}\frac{2N_D Q}{n_1 T}}$$

Čomu zodpovedajú optimálne náklady:

$$N(x_O, s_O) = \sqrt{\frac{n_2}{n_1 + n_2} \cdot 2 \cdot Q \cdot T \cdot N_D \cdot n_1}$$

Príklad

Strojárenský podnik potrebuje ročne 100 000 ks súčiastok určitého typu. Náklady na jednu dodávku sú 2000 PJ. Ročné skladovacie náklady pre tieto súčiastky sú 4 PJ/ks/rok. Pri nedostatku súčiastok na sklade vzniknú dodatočné jednotlové náklady 12 PJ/ks/rok. Koľko súčiastok a ako často má podnik objednať?

Riešenie

Zo zadania úlohy vyplýva, že Q = 100 000 ks/rok, N_D = 2000 PJ/obj., n_1 = 4 PJ/ks/rok a n_2 = 12 PJ/ks/rok. Takže dosadením do vzťahov pre výpočet optimálnej veľkosti objednávaných zásob a signálnej úrovne zásob a ich postupným vyčíslením dostávame:

$$x_0 = \sqrt{\frac{4+12}{12}.\frac{2.2000.100000}{1.4}} = 11547$$

$$s_0 = \sqrt{\frac{12}{4+12}.\frac{2.2000.100000}{1.4}} = 8660$$

Takže optimálny interval dopĺňania zásob bude:

$$t = 1.\frac{11547}{100000} = 0{,}11547 = 42.dní$$

8 Použitá literatúra

[Aggoun, Beldiceaunu 1993] Aggoun, A. and Beldiceanu, N.: Extending CHIP in Order to Solve Complex Scheduling and Placement Problems, *Math. Comut. Modelling*, Vol. 17, No. 7, pp. 57-73, 1993.

[Apt, Wallace 2006] Apt Krzysztof R., Wallace M.: Constraint Logic Programming using Eclipse. ISBN-13: Cambridge University Press, 2006, 329 s.

[Beldiceau, Contejean 1994] Beldiceau, N. and Contejean, E.: Introducing Global Constraints in CHIP, *Math. Comut. Modelling*, Vol. 20, No. 12, 1994, pp. 97-123.

[Benhamou 1993] Benhamou, F.: Boolean Algorithms in Prolog III, in *Constraint Logic Programming, Selected Research*, ed. by Benhamou, F., Colmerauer, A., MIT Press 1993, pp. 307-325.

[Blazewicz et al. 1996] Blazewicz J., Ecker K.H., Pesch E., Schmidt G., Weglarz J.: Scheduling Computer and manufacturing Processes. Springer 1996, ISBN 3-540-61496-6, 491 s.

[Caseau, Laburthe 1995] Caseau, Y. and Laburthe, F.: Disjunctive Scheduling with Task Intervals. LIENS Technical Report 95-25, Laboratoire d'Informatique de I'Ecole Normale Superieure, 1995

[Cras 1993] Cras, J. Y.: A Review of Industrial Constraint solving Tools. A report in *AI Perspective Series*, Oxford, United Kingdom, 1993.

[Dechter 1992] Dechter, R.: Constraint Networks, in *Encyclopedia of Artificial Intelligence*, 2nd edition, Wiley and Sons, 1992, pp 276-285.

[Dincbas et al. 1988] Dincbas, M., Van Hentenryck, P., Simonis, H., Aggoun, A., Graf, T. and Berthier, F.: The Constraint Logic Programming Language CHIP. In *Proc. of the International Conference on 5th Generation Computer Systems*, ICOT, 1988.

[Dudorkin 1993] Dudorkin, J.: Operační výzkum.Vydavatelství ČVÚT Praha, 1997, ISBN 80-01-01571-8, 295 s.

[Dupaľ, Brezina 2005] Dupaľ, A., Brezina, I.: Logistika v manažmente podniku. Sprint, Bratislava, 2005, ISBN 80-89085-38-5, 326 s.

[Fox 1987] Fox, M.: *Constraint-Directed Search: A Case Study of Job-Shop Scheduling*, Morgan Kaufman, 1987.

[Jánošíková 1994] Jánošíková, Ľ.: Optimalizačné úlohy na dopravných sieťach a metóda tabu search. Kandidátska dizertačná práca, Vysoká škola dopravy a spojov v Žiline, 1994.

[Kirkpatrick et al. 1983] Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P.: Optimization by Simulated Annealing. *Science*, No. 220, 1983, pp. 671-680.

[Lambert et al. 2000] Lambert, D., Stock, J.R., Ellram, L. 2000. Logistika. Computer Press Praha, ISBN 80-7226-221-1, 589 s.

[Mach 1996] Mach, M.: Using Genetic Algorithms for Problems with Soft Constraints. In *Proc. of the 3rd Workshop on Artificial Intelligence Techniques AIT'96*, Brno, 1996, pp. 173-174.

[Mach, Paralič 2000] Mach, M., Paralič, J. 2000: Úlohy a ohraničeniami – od teórie k programovaniu. Elfa, 2000, ISBN 80-88964-48-2, 217 s.

[Madarász et al. 2008] Madarász L., Bučko M., Andoga R.: Systémová analýza a syntéza. Edícia učebných textov Fakulty elektrotechniky a informatiky, TU v Košiciach, Elfa, 2008, ISBN 978-80-8086-080-6, 266 s.

[Malindžák 1996] Malindžák, D.: Výrobná logistika I. FBERG, Technická univerzita v Košiciach, Vydavateľstvo Štrofek, Košice 1996, ISBN 80-967325-1-X, 165 s.

[Muth, Thomson 1963] Muth, J. and Thomson, G.: *Industrial Scheduling*. Prentice Hall, 1963.

[Paralič 1995] Paralič, J.: Constraint Logic Programming - An Overview. Rigorózna práca, ECRC Mníchov - Technická univerzita v Košiciach, Jún 1995.

[Paralič 1997] Paralič J.: Riešenie úloh rozvrhovania logickým programovaním ohraničení. Dizertačná práca, Katedra kybernetiky a umelej inteligencie, FEI, Technická univerzita v Košiciach, máj 1997, 114 s.

[Paralič 2003] Paralič J.: Objavovanie znalostí v databázach. Technická univerzita v Košiciach, vydavateľstvo Elfa, ISBN 80-89066-60-7, 80 s.

[Paralič, Sabol 1995] Paralič J., Sabol T.: Analysis of the Constraint Satisfaction Algorithms with respect to Configuration Design. Technical Report TR-Encode-TUKE-2-95, Technical University of Košice, 1995.

[Puget 1994] Puget, J.F.: A C++ Implementation of CLP, Technical Report 94-01, ILOG S.A., Gentilly, France, 1994.

[Tsang 1993] Tsang, E.: *Foundations of Constraint Satisfaction*. Academic Press, 1993.

[Tsang 1995] Tsang, E.: Scheduling Techniques - A Comparative Study. In *BT Technology Today*, Vol.13, No.1, 1995, pp. 16-28.

Autor: Ján Paralič
Názov: Rozvrhovanie a logistika
Vydanie: prvé
Náklad: 80
Rozsah: 94
Vydavateľ: Equilibria, s.r.o., Košice
Formát: B5
Tlač: Equilibria, s.r.o., Košice
Vytlačené: máj 2010

ISBN