UNIVERZITA PALACKÉHO V OLOMOUCI
PŘÍRODOVĚDECKÁ FAKULTA
KATEDRA MATEMATICKÉ ANALÝZY A APLIKACÍ MATEMATIKY

BAKALÁŘSKÁ PRÁCE

Distribuční úloha a jiná zobecnění dopravního problému

Vedoucí bakalářské práce:
RNDr. Pavel Ženčák, Ph.D.
Rok odevzdání: 2013

Vypracovala:
Jana Falaštová
MAP, III. ročník

Prohlášení

Prohlašuji, že jsem bakalářskou práci zpracovala samostatně pod vedením RNDr. Pavla Ženčáka, Ph.D. a v seznamu literatury jsem uvedla všechny použité zdroje.

V Olomouci dne 8. dubna 2013 Jana Falaštová v.r.

Poděkování

Děkuji všem, kteří přispěli k tvorbě této bakalářské práce, především RNDr. Pavlu Ženčákovi, Ph.D. za všechny poskytnuté rady, užitečné připomínky a informace a velkou trpělivost. Dále děkuji své mamince a příteli, že mě po celou dobu studia podporovali.

Obsah

Úvod

Cílem této práce je seznámit se s metodami pro řešení dopravní úlohy a distribuční úlohy (nazývané také vážený distribuční problém [6] nebo obecný distribuční problém [5]). Tyto problémy patří do lineárního programování, které je speciálním případem konvexního programování. To se využívá v různých oborech při optimalizaci, například v ekonomii, zemědělství, průmyslu a dopravě. Teoretické základy lineárnímu programování dal na přelomu 19. a 20. století matematik Farkas s teorií soustav lineárních nerovností. Úlohy lineárního programování se nejčastěji řeší simplexovou metodou. Tato metoda byla v roce 1947 uvedena G. D. Dantzigem pro řešení úloh lineárního programování pro americké letectvo (viz [2] nebo [6]). Vedle simplexové metody se objevily další metody, například v roce 1979 elipsoidová metoda a v roce 1984 metoda vnitřních bodů, která jediná dokáže simplexové metodě konkurovat v reálných případech. Metody pro řešení speciálních úloh vychází ze simplexové metody, ale využívají speciálního struktury úloh k urychlení a zjednodušení výpočtu.

Nejdříve si v první kapitole uvedeme několik základních poznatků z lineárního programování, základní definice a poznatky z duality, které použijeme při odvozování metod.

V druhé kapitole se budeme zabývat dopravní úlohou, která je někdy nazývaná Hitchcockova podle autora F.L.Hitchcocka, který v roce 1941 vytvořil obecnou metodu pro její řešení. Je to úloha o přepravě zboží od dodavatelů k odběratelům přímou cestou. Ukážeme si metody pro nalezení řešení, které si ihned názorně ukážeme na konkrétním příkladě.

Ve třetí kapitole si uvedeme distribuční úlohu, která je zobecněním dopravní úlohy. Úloha slouží především pro optimalizaci výroby. Uvedeme si metody pro nalezení řešení, následně si je vyzkoušíme pro názornost na konkrétním příkladě. Pro každou metodu jsem naprogramovala funkce v matematickém softwaru Matlab. Výpisy kódů funkcí uvedeme v přílohách. Uvedeme si také výsledky při použití těchto funkcí na konkrétních příkladech.

V poslední kapitole si uvedeme několik dalších speciálních úloh lineárního programování.

1 Úvod do lineárního programování

Úlohy, kterými se zde budeme zabývat jsou speciálními případy lineárního programování, a proto si nejprve uvedeme některé známé pojmy a výsledky z této oblasti. Každá úloha lineárního programování spočívá v nalezení extrému (minima nebo maxima) lineární funkce, jejíž proměnné jsou vázány lineárními podmínkami. Uveďme si nejprve základní poznatky z lineárního programování.

1.1 Obecná úloha lineárního programování

Každá úloha lineárního programování lze obecně matematicky zapsat následovně

$$\begin{aligned} &\text{minimalizovat } \sum_{i=1}^{m} c_i x_i \\ &\text{za podmínek } \sum_{i=1}^{m} d_{i,j} x_i \leq a_j && j = 1, 2, ..., k;\ k \leq n \\ &\qquad\qquad \sum_{i=1}^{m} d_{i,j} x_i = a_j && j = k+1, k+2, ..., n \\ &\qquad\qquad x_i \geq 0 && i \in \mathcal{I}_1; \quad \mathcal{I}_1 \subset \{1, 2, ..., m\} \end{aligned} \tag{1.1}$$

Úlohu můžeme zapsat i ve vektorovém tvaru

$$\begin{aligned} &\text{minimalizovat } \mathbf{c^T x} \\ &\text{za podmínek } \mathbf{d^j x} \leq a_j && j = 1, 2, ..., k;\ k \leq n \\ &\qquad\qquad \mathbf{d^j x} = a_j && j = k+1, k+2, ..., n \\ &\qquad\qquad x_i \geq 0 && i \in \mathcal{I}_1; \quad \mathcal{I}_1 \subset \{1, 2, ..., m\} \end{aligned}$$

Kde $\mathbf{c}$ a $\mathbf{x}$ jsou sloupcové vektory $m \times 1$, a_j jsou složky vektoru pravých stran $\mathbf{a}$, který je typu $n \times 1$ a $\mathbf{d^j}$ jsou jednotlivé řádky matice $\mathbf{D}$, jejíž prvky jsou koeficienty $d_{i,j}$ a je typu $n \times m$.

Pokud jsou všechny podmínky ve tvaru rovnosti, řekneme, že úloha je v rovnicovém tvaru.

$$\begin{aligned} &\text{minimalizovat } \sum_{i=1}^{m} c_i x_i \\ &\text{za podmínek } \sum_{i=1}^{m} d_{i,j} x_i = a_j && j = 1, 2, ..., n; \\ &\qquad\qquad x_i \geq 0 && i \in \{1, 2, ..., m\} \end{aligned} \tag{1.2}$$

Lze ji také zapsat v maticovém tvaru:

$$\begin{aligned} &\text{minimalizovat } \mathbf{c^T x} \\ &\text{za podmínek } \mathbf{Dx} = \mathbf{a} \\ &\qquad\qquad\quad \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{1.3}$$

Poznámka 1.1. Úlohu ve tvaru (1.1) lze převést na úlohu ve tvaru (1.2) jelikož:

1. Každou podmínku ve tvaru nerovnosti lze převést na podmínku ve tvaru rovnosti zavedením tzv. doplňkových proměnných. Podmínku ve tvaru

$$\sum_{i=1}^{m} d_{i,j} x_i \leq a_j \quad j = 1, 2, ..., k;\ k \leq n$$

převedeme na podmínku

$$\sum_{i=1}^{m} d_{i,j} x_i + x_{m+1} = a_j \quad x_{m+1} \geq 0;\ j = \{1, 2, ..., k\};\ k \leq n$$

2. Každou proměnnou $x_i \notin \mathcal{I}_1$ lze zapsat ve tvaru $x_i = x_i^+ - x_i^-$, kde $x_i^+ = \max\{x_i, 0\} \geq 0$ a $x_i^- = \max\{-x_i, 0\} \geq 0$.

Poznámka 1.2. Jelikož platí $\min f(\mathbf{x}) = -\max(-f(\mathbf{x}))$ můžeme se zabývat pouze minimalizační úlohou [2].

1.2 Základní pojmy

Nyní si nadefinujeme základní pojmy a věty, se kterými se dále budeme setkávat. Viz [3] a [2].

Definice 1.3. Přípustným řešením úlohy lineárního programování nazýváme takový vektor $\mathbf{x} = (x_1, x_2, ..., x_m)^T$, který vyhovuje podmínkám úlohy (1.2). Množinu přípustných řešení označme $M = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{Dx} = \mathbf{a}, \mathbf{x} \geq \mathbf{0}\}$.

Definice 1.4. Přípustné řešení $\mathbf{x} = (x_1, x_2, ..., x_m)^T$ úlohy (1.3) nazveme **přípustné bázové řešení**, jestliže sloupce matice $\mathbf{D}$ odpovídající kladným složkám vektoru $\mathbf{x}$ jsou lineárně nezávislé. Tyto kladné složky vektoru $\mathbf{x}$ nazveme báze. Matici těchto vektorů odpovídajících kladným složkám nazveme bázová matice.

Definice 1.5. Optimální řešení je takové přípustné řešení $\mathbf{x}^*$, které minimalizuje danou úlohu, tedy platí $\mathbf{c^T x^* \leq c^T x}$, pro všechna $\mathbf{x}$ z množiny přípustných řešení.

Věta 1.6. *(O existenci optimálního řešení)*
Nechť množina přípustných řešení úlohy (1.3) *je neprázdná tj.* $M = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{Dx} = \mathbf{a}, \mathbf{x} \geq \mathbf{0}\} \neq \emptyset$. *Jestliže existuje reálné číslo* γ *takové, že pro libovolné* $\mathbf{x} \in M$ *platí* $\mathbf{c}^T\mathbf{x} \geq \gamma$, *pak existuje optimální řešení úlohy lineárního programování minimalizovat* $\mathbf{c}^T\mathbf{x}$ *na množině* M.

Důkaz. Viz [2] □

Dále si uvedeme základní větu lineárního programování převzatou z [2].

Věta 1.7. *(Základní věta lineárního programování)*
Pro úlohu lineárního programování (1.3), *neboli minimalizovat* $\mathbf{c}^T\mathbf{x}$ *na množině* $M = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{Dx} = \mathbf{a}, \mathbf{x} \geq \mathbf{0}\}$, *platí jedna ze tří možností:*

1. $M = \emptyset$
2. $M \neq \emptyset$ *a* $\inf\limits_{\mathbf{x} \in M} \mathbf{c}^T\mathbf{x} = -\infty$ *(tj.* $M^* = \{\mathbf{x} \in M \mid \mathbf{c}^T\mathbf{x} = \min\limits_{\mathbf{z} \in M} \mathbf{c}^T\mathbf{z}\} = \emptyset$*)*
3. $M^* \neq \emptyset$

Navíc:
(a) Je-li $M \neq \emptyset$, *existuje bázové přípustné řešení.*
(b) Je-li $M^* \neq \emptyset$, *existuje bázové optimální řešení.*

Lagrangeova funkce a Karush-Kuhn-Tackerovy podmínky

Zavedeme si Lagrangeovu funkci. Tato funkce je důležitá k vytvoření duální úlohy, a tedy k nalezení optimálního řešení úlohy konvexního programování. Napišme ji pro danou úlohu (1.1) ve tvaru

$$L(x, u, s) = \sum_{i=1}^{m} c_i x_i + \sum_{j=1}^{n} u_j (a_j - \sum_{i=1}^{m} d_{i,j} x_i) - \sum_{i \in I_1} s_i x_i \tag{1.4}$$

kde u_j jsou Lagrangeovy multiplikátory pro obecné podmínky a s_i jsou Lagrangeovy multiplikátory pro jednoduché podmínky (tj. podmínka $x_i \geq 0$). Viz [4]

Věta 1.8. *Nechť bod x^* je bod lokálního řešení úlohy. Pak existuje vektor Lagrangeových multiplikátorů* $\mathbf{u}$*, pro který platí následující podmínky:*

$$\begin{aligned}
\nabla_{\mathbf{x}} L(\mathbf{x}^*, \mathbf{u}, \mathbf{s}) &= 0 & \\
\mathbf{d^j x}^* &\leq a_j & j = 1, 2, ..., k \\
\mathbf{d^j x}^* &= a_j & j = k+1, k+2, ..., n \\
u_j &\leq 0 & j = 1, 2, ..., k \\
s_i &\geq 0 & i \in I_1 \\
u_j \cdot \left(a_j - \mathbf{d^j x}^*\right) &= 0 & j = 1, 2, ..., n \\
x_i s_i &= 0 & i \in I_1
\end{aligned}$$

Tyto podmínky se nazývají Karush-Kuhn-Tackerovy podmínky (zkráceně jim budeme říkat KKT podmínky). Poslední podmínky $u_j \cdot \left(a_j - \mathbf{d^j x}^*\right) = 0$ a $x_i s_i = 0$ se nazývají podmínky komplementarity. Viz [1], [4].

Duální úloha

Úlohu (1.1) uvedenou na straně 5 budeme nyní nazývat úloha primární. Každé úloze nejen lineárního, ale obecně konvexního programování lze přiřadit duální úlohu.

Pro tvorbu duální úlohy lineárního programování můžeme využít toho, že je speciální úlohou konvexního programování. Pomocí Lagrangeovi funkce si nejprve zformulujeme Wolfeho duální úlohu k úloze (1.1):

$$\begin{aligned}
\text{maximalizovat} \quad & L(\mathbf{x}, \mathbf{u}, \mathbf{s}) & \\
\text{za podmínek} \quad & \nabla_x L(\mathbf{x}, \mathbf{u}, \mathbf{s}) = 0 & \\
& \mathbf{s} \geq 0 & \\
& u_j \leq 0 & j = 1, 2, ..., k
\end{aligned} \tag{1.5}$$

Nyní jsou u_j duální proměnné a s_i doplňkové duální proměnné. Po přepsání a upravení

se úloha zjednoduší na

$$\begin{aligned}
\text{maximalizovat} \quad & \sum_{j=1}^{n} a_j u_j & & \\
\text{za podmínek} \quad & \sum_{j=1}^{n} d_{i,j} u_j + s_i = c_i & & i \in I_1 \\
& \sum_{j=1}^{n} d_{i,j} u_j = c_i & & i \in \{1, 2, ..., m\} \setminus I_1 \\
& s_i \geq 0 & & i \in I_1 \\
& u_j \leq 0 & & j = 1, 2, ..., k
\end{aligned}$$

Jelikož duální proměnné s_i mají charakter doplňkových proměnných, můžeme jejich vynecháním dostat úlohu ve tvaru, který je typický pro lineární programování.

$$\begin{aligned}
\text{maximalizovat} \quad & \sum_{j=1}^{n} a_j u_j & & \\
\text{za podmínek} \quad & \sum_{j=1}^{n} d_{i,j} u_j \leq c_i & & i \in I_1 \\
& \sum_{j=1}^{n} d_{i,j} u_j = c_i & & i \in \{1, 2, ..., m\} \setminus I_1 \\
& u_j \leq 0 & & j = 1, 2, ..., k
\end{aligned} \tag{1.6}$$

Vztahy mezi duální a primární úlohou

Pro primární a duální úlohu lineárního programování platí určité vztahy. Lehce se dá ověřit, že duální úloha k duální úloze je primární úloha. Jedná se tedy o dvojici vzájemně duálních úloh (viz [2]). Záleží, ze které úlohy vyjdeme, tedy kterou si označíme jako primární.

Věta 1.9. *Má-li primární nebo duální úloha optimální řešení, pak má optimální řešení i druhá a přitom platí*

$$\min \sum_{i=1}^{m} c_i x_i = \max \sum_{j=1}^{n} a_j u_j$$

Jestliže účelová funkce jedné z úloh (primární nebo duální) není ohraničená (může růst nebo klesat neomezeně), pak druhá úloha nemá přípustné řešení.

Důkaz. viz [3] □

Věta 1.10. *Nemá-li primární či duální úloha přípustné řešení, nemá ani jedna z nich optimální řešení.*

Důkaz. viz [2] □

1.3 Obecná distribuční úloha a její duální úloha

Obecnou distribuční úlohu zapíšeme takto:

$$\begin{aligned} \text{minimalizovat} \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{i,j}x_{i,j} & & \\ \text{za podmínek} \quad & \sum_{j=1}^{n} e_{i,j}x_{i,j} \leq a_i & & i = 1,2,...,m \\ & \sum_{i=1}^{m} d_{i,j}x_{i,j} = b_j & & j = 1,2,...,n \\ & x_{i,j} \geq 0 & & i = 1,2,...,m;\ j = 1,2,...,m \end{aligned} \tag{1.7}$$

Jelikož jsou zde podmínky ve speciálním tvaru, budeme zde rozlišovat duální proměnné u_i pro podmínky ve tvaru nerovnosti a duální proměnné v_j pro podmínky ve tvaru rovnosti. Duální úloha má následující tvar

$$\begin{aligned} \text{maximalizovat} \quad & \sum_{i=1}^{m} a_i u_i + \sum_{j=1}^{n} b_j v_j & & \\ \text{za podmínek} \quad & e_{i,j}u_i + d_{i,j}v_j \leq c_{i,j} & & i = 1,2,...,m; j = 1,2,...,n \\ & u_i \leq 0 & & i = 1,2,...,m \end{aligned} \tag{1.8}$$

2 Dopravní úloha

Dopravní úlohou rozumíme problém, kde máme m dodavatelů (např. skladů), kteří mají k dispozici zboží v množství $a_1, a_2, ..., a_m$. Zboží se má přímou cestou rozvést mezi n odběratelů (např. obchodů), kde každý odběratel požaduje zboží v množství $b_1, b_2, ..., b_n$. Požadujeme minimální celkové náklady na dopravu, přičemž $c_{i,j}$ je cena dopravy za jednotku zboží mezi i-tým dodavatelem a j-tým odběratelem. Problém můžeme zapsat matematicky takto:

$$\begin{aligned} \text{minimalizovat} \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{i,j}x_{i,j} & & \\ \text{za podmínek} \quad & \sum_{j=1}^{n} x_{i,j}{=}a_i & & i = 1, 2, ..., m \\ & \sum_{i=1}^{m} x_{i,j} = b_j & & j = 1, 2, ..., n \\ & x_{i,j} \geq 0 & & i = 1, 2, ..., m; j = 1, 2, ..., n \end{aligned} \tag{2.1}$$

kde $c_{i,j} \geq 0$, $a_i \geq 0$ a $b_j \geq 0$ pro $i = 1, 2, ..., m$ a $j = 1, 2, ..., n$. Proměnné $x_{i,j}$ označují množství přepravovaného zboží mezi i-tým dodavatelem a j-tým odběratelem.

Úlohu si můžeme zapsat i maticově

$$\begin{aligned} \min \mathbf{c^T x} \\ \text{za podmínek } \widetilde{\mathbf{A}}\mathbf{x} = \widetilde{\mathbf{b}} \\ \mathbf{x} \geq \mathbf{0} \end{aligned}$$

kde

$$\begin{aligned} \mathbf{c} &= (c_{1,1}, c_{1,2}, ..., c_{1,n}, c_{2,1}, ..., c_{m,n})^T \\ \mathbf{x} &= (x_{1,1}, x_{1,2}, ..., x_{1,n}, x_{2,1}, ..., x_{m,n})^T \\ \widetilde{\mathbf{b}} &= (a_1, a_2, ..., a_m, b_1, b_2, ..., b_n)^T \end{aligned}$$

Jelikož dopravní úloha je pouze speciální případ lineárního programování má matice

$\widetilde{\mathbf{A}}$ speciální tvar.

$$\widetilde{\mathbf{A}} = \left(\begin{array}{cccc|cccc|ccc|cccc} 1 & 1 & \cdots & 1 & & & & & & & & & & & \\ & & & & 1 & 1 & \cdots & 1 & & & & & & & \\ & & & & & & & & & \ddots & & & & & \\ & & & & & & & & & & & 1 & 1 & \cdots & 1 \\ \hline - & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - \\ \hline 1 & & & & 1 & & & & & & & 1 & & & \\ & 1 & & & & 1 & & & & \cdots & & & 1 & & \\ & & \ddots & & & & \ddots & & & & & & & \ddots & \\ & & & 1 & & & & 1 & & & & & & & 1 \end{array}\right)$$

Matice $\widetilde{\mathbf{A}}$ má $(m+n)$ řádků ale pouze $(m+n-1)$ je jich lineárně nezávislých [2].

Při ručním řešení nepracujeme s maticovou formulací, ale zapisujeme problém do následující tabulky.

$x_{1,1}$ $c_{1,1}$	$x_{1,2}$ $c_{1,2}$	...	$x_{1,n}$ $c_{1,n}$	a_1
$x_{2,1}$ $c_{2,1}$	$x_{2,2}$ $c_{2,2}$	...	$x_{2,n}$ $c_{2,n}$	a_2
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$x_{m,1}$ $c_{m,1}$	$x_{m,2}$ $c_{m,2}$	...	$x_{m,n}$ $c_{m,n}$	a_m
b_1	b_2	$\cdots$	b_n	

2.1 Řešitelnost dopravní úlohy

Můžeme rozlišit dva typy dopravní úlohy, vyrovnaný a nevyrovnaný. Určení typu úlohy má vliv na řešitelnost.

Vyrovnaný dopravní problém je takový problém, pro který platí dodavetelsko-odběratelská rovnováha, tj. platí $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$. Tento problém nemusíme nijak upravovat a počítáme podle postupu, který si popíšeme později.

Nevyrovnaný dopravní problém je takový problém, kde $\sum_{i=1}^{m} a_i \neq \sum_{j=1}^{n} b_j$.

Věta 2.1. *Pro dopravní úlohu* (2.1) *jsou následující podmínky ekvivalentní:*

a)má přípustné řešení

b)má optimální řešení

c)platí dodavatelsko-odběratelská rovnováha, tj. platí $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$

Důkaz. viz [2] □

Poznámka. Aplikujeme-li základní větu lineárního programování (1.7) na dopravní úlohu, zjistíme, že oproti obecné úloze lineárního programování u dopravní úlohy nemůže nastat 2.možnost věty (1.7) jelikož $c_{i,j}$ i $x_{i,j}$ jsou zdola omezené nulou. Vždy může nastat 3.možnost. Což znamená, že platí-li dodavateslko-odběratelská rovnováha, existuje vždy optimální řešení, což je ve zkratce tvrzení věty (2.1).

Díky předchozí větě nám stačí ověřit zda je problém vyrovnaný. Je-li vyrovnaný, má přípustné i optimální řešení. Máme-li problém nevyrovnaný, tak podle předchozí věty tento problém nemá přípustné ani optimální řešení. Ovšem i tento nevyrovnaný problém můžeme řešit, ale zohledníme to v prezentaci výsledku. Rozlišujeme dva případy nevyrovnaného problému:

1. Platí-li $\sum_{i=1}^{m} a_i < \sum_{j=1}^{n} b_j$, původní úloha nemá řešení. Lze ovšem hledat řešení úlohy

$$\begin{aligned} \text{minimalizovat} \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{i,j}x_{i,j} & \\ \text{za podmínek} \quad & \sum_{j=1}^{n} x_{i,j}=a_i & i=1,2,...,m \\ & \sum_{i=1}^{m} x_{i,j} \leq b_j & j=1,2,...,n \\ & x_{i,j} \geq 0 & i=1,2,...,m; j=1,2,...,n \end{aligned}$$

   Abychom úlohu převedli na rovnicový tvar, musíme doplnit fiktivního dodavatele s množstvím $a_{m+1} = \sum_{j=1}^{n} b_j - \sum_{i=1}^{m} a_i$ a s nulovými přepravními cenami $c_{m+1,j} = 0$ pro $j = 1, 2, ...n$.

2. Platí-li $\sum_{i=1}^{m} a_i > \sum_{j=1}^{n} b_j$, původní úloha nemá řešení ale lze řešit úloha:

$$\begin{aligned} \text{mininalizovat} \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{i,j}x_{i,j} & \\ \text{za podmínek} \quad & \sum_{j=1}^{n} x_{i,j} \leq a_i & i=1,2,...,m \\ & \sum_{i=1}^{m} x_{i,j} = b_j & j=1,2,...,n \\ & x_{i,j} \geq 0 & i=1,2,...,m; j=1,2,...,n \end{aligned}$$

   Podobně jako u případu 1. doplníme fiktivního odběratele s požadavkem $b_{n+1} = \sum_{i=1}^{m} a_i - \sum_{j=1}^{n} b_j$ a s nulovými cenami $c_{i,n+1} = 0 \quad i = 1, 2, ..., m$.

Přidáním doplňkových proměnných (fiktivního dodavatele nebo odběratele) převedeme problém na vyrovnaný a můžeme jej řešit stejným způsobem jakoby byl problém vyrovnaný od počátku. Při prezentaci výsledku řešení musíme brát v úvahu, že jsme původně řešili nevyrovnaný problém.

Příklad 2.2. Máme úlohu zadanou tabulkou, kde máme 3 výrobce nábytku a 3 obchody. První výrobce má k dispozici 60 kusů židlí, druhý výrobce 40 kusů a třetí výrobce 55 kusů. První obchod má objednávku na 65 kusů, druhý obchod na 55 kusů a třetí na 35 kusů. Přepravní ceny $c_{i,j}$ za jednotku zboží máme zapsány v tabulce.

$c_{i,j}$	obchod 1	obchod 2	obchod 3	množství
výrobce 1	3	2	2	60
výrobce 2	1	4	3	40
výrobce 3	5	2	4	55
množství	65	55	35	

Nejprve je třeba ověřit, zda je problém vyrovnaný. Celkové množství zboží u výrobců je $60 + 40 + 55 = \underline{155}$ a celkové množství požadované obchody je $65 + 55 + 35 = \underline{155}$. Problém je vyrovnaný a tedy podle věty 2.1 má úloha přípustné i optimální řešení a můžeme ji bez obav řešit. Příklad si můžeme zapsat i matematicky

$$
\begin{aligned}
\text{minimalizovat } & 3x_{1,1} + 2x_{1,2} + 2x_{1,3} + 1x_{2,1} + 4x_{2,2} + 3x_{2,3} + 5x_{3,1} + 2x_{3,2} + 4x_{3,3} \\
\text{za podmínek } & x_{1,1} + x_{1,2} + x_{1,3} = 60 \\
& x_{2,1} + x_{2,2} + x_{2,3} = 40 \\
& x_{3,1} + x_{3,2} + x_{3,3} = 55 \\
& x_{1,1} + x_{2,1} + x_{3,1} = 65 \\
& x_{1,2} + x_{2,2} + x_{3,2} = 55 \\
& x_{1,3} + x_{2,3} + x_{3,3} = 35 \\
& x_{i,j} \geq 0 \quad \text{pro } i = 1,2,3 \text{ a } j = 1,2,3
\end{aligned}
$$

Algoritmy pro hledání řešení dopravní úlohy si pro názornost budeme předvádět na tomto příkladu.

2.2 Nalezení počátečního bázového řešení

Při hledání optimálního řešení dopravního problému nejprve nalezneme počáteční bázové řešení a poté jej dále upravujeme až dostaneme optimální řešení. Tyto základní metody můžeme nalézt v [7], [2] nebo [5].

Nejdříve si uvedeme obecný algoritmus pro nalezení počátečního řešení, který neurčuje postup řešení jednoznačně.

Algoritmus 2.3. *Mějme indexové množiny* $\mathcal{I}=\{1,2,...,m\}$ *a* $\mathcal{J}=\{1,2,...,n\}$.

1. *Zvolíme libovolné indexy* $i \in \mathcal{I}$ *a* $j \in \mathcal{J}$. *Hodnotu prvku s těmito indexy položíme* $x_{i,j} = \min\{a_i, b_j\}$ *a hodnoty* a_i *a* b_j *přepočítáme užitím vzorců* $a_i = a_i - x_{i,j}$ *a* $b_j = b_j - x_{i,j}$.

2. *Z indexů* i *nebo* j *vynecháme z příslušné množiny jeden index a to ten, pro který se vynulují požadavky (tj.* $a_i = 0$ *nebo* $b_j = 0$*). V tabulce to znamená, že vyškrtneme příslušný řádek či sloupec. Vždy vynecháme pouze jeden z indexů. V případě, že se vynulují hodnoty pro oba indexy, vynecháme index přednostně z množiny, která obsahuje alespoň dva indexy, pokud taková množina existuje. Tím dostáváme nové indexové množiny.*

3. *Celý postup opakujeme dokud nevyčerpáme všechny indexy* i *a* j.

Poznámka. Nebázové prvky (nebázové nuly) do tabulky nezapisujeme.

Nyní si uvedeme speciální algoritmy zpřesňující postup obecného algoritmu 2.3. V obecném algoritmu jsme volili indexy i a j libovolně, následující metody nám jasně určují, jak máme indexy i a j volit.

2.2.1 Metoda severozápadního rohu

Tato metoda je speciální případ obecného algoritmu, kde indexy i a j volíme vždy nejmenší možné (tj. začínáme od levého horního rohu a směřujeme do pravého dolního rohu tabulky).

Příklad. Ukážeme si postup na příkladě 2.2. V prvním kroku máme indexní množiny $\mathcal{I} = \{1,2,3\}$ a $\mathcal{J} = \{1,2,3\}$, zvolíme $i = 1$ a $j = 1$, $x_{11} = \min\{60;65\} = 60$. Přepočítáme $a_1 = 0$ a $b_1 = 5$.

$x_{i,j}$ $c_{i,j}$	obchod 1	obchod 2	obchod 3	
výrobce 1	60 ^{3}	^{2}	^{2}	0
výrobce 2	^{1}	^{4}	^{3}	40
výrobce 3	^{5}	^{2}	^{4}	55
	5	55	35	

Protože pro $i = 1$ se hodnota a_1 nuluje, tento index vynecháme. Máme tedy indexní množiny $\mathcal{I} = \{2,3\}$ a $\mathcal{J} = \{1,2,3\}$, pokračujeme s $i = 2$ a $j = 1$, kde volíme $x_{2,1} = 5$ a přepočítáme $a_2 = 35$ a $b_1 = 0$.

$x_{i,j}$ $c_{i,j}$	obchod 1	obchod 2	obchod 3	
výrobce 1	60 ^{3}	^{2}	^{2}	0
výrobce 2	5 ^{1}	^{4}	^{3}	35
výrobce 3	^{5}	^{2}	^{4}	55
	0	55	35	

Protože hodnota b_1 se nuluje, vynecháváme index $j = 1$. Máme tedy $\mathcal{I} = \{2, 3\}$ a $\mathcal{J} = \{2, 3\}$ a pokračujeme indexy $i = 2$ a $j = 2$, kde $x_{2,2} = 35$. Přepočítáme a_2 a b_2.

$x_{i,j}$ $c_{i,j}$	obchod 1	obchod 2	obchod 3	
výrobce 1	60 ^{3}	^{2}	^{2}	0
výrobce 2	5 ^{1}	35 ^{4}	^{3}	0
výrobce 3	^{5}	^{2}	^{4}	55
	0	20	35	

Tímto postupem pokračujeme dále, až nalezneme počáteční řešení, které si zapíšeme do původní tabulky.

$x_{i,j}$ $c_{i,j}$	obchod 1	obchod 2	obchod 3	
výrobce 1	60 ^{3}	^{2}	^{2}	60
výrobce 2	5 ^{1}	35 ^{4}	^{3}	40
výrobce 3	^{5}	20 ^{2}	35 ^{4}	55
	65	55	35	

Celková cena $\sum_{i=1}^{3} \sum_{j=1}^{3} x_{i,j} c_{i,j} = 3 \cdot 60 + 1 \cdot 5 + 4 \cdot 35 + 2 \cdot 20 + 4 \cdot 35 = 505$

2.2.2 Metoda minimální ceny

V této metodě jsou přednostně obsazovány políčka s nejnižší cenou. To znamená, že v tabulce nalezneme nejmenší hodnotu $c_{i,j}$ a toto pole obsadíme $x_{i,j} = \min\{a_i, b_j\}$. Existuje-li více polí se stejnou minimální cenou, volíme libovolné z nich.

Příklad. Pro náš konkrétní příklad 2.2 máme opět indexní množiny $\mathcal{I} = \{1, 2, 3\}$ a $\mathcal{J} = \{1, 2, 3\}$. Nejnižší cena je v poli s indexy $i = 2$ a $j = 1$. Obsadíme jej hodnotou $x_{2,1} = \min\{40; 65\} = 40$. Přepočítáme $a_2 = 0$ a $b_1 = 25$.

$x_{i,j}$ $c_{i,j}$	obchod 1	obchod 2	obchod 3	
výrobce 1	^{3}	^{2}	^{2}	60
výrobce 2	40 ^{1}	^{4}	^{3}	0
výrobce 3	^{5}	^{2}	^{4}	55
	25	55	35	

Dostaneme nové indexní množiny $\mathcal{I} = \{1, 3\}$ a $\mathcal{J} = \{1, 2, 3\}$. Dále máme hned tři pole se stejnou nejnižší cenou $x_{1,2}$, $x_{1,3}$ a $x_{3,2}$ vybereme tedy libovolné z nich, např. první z nich $x_{1,2}$ a položíme rovnu $x_{1,2} = \min\{60; 55\} = 55$. Nyní přepočteme hodnoty $a_1 = 5$ a $b_2 = 0$.

$x_{i,j}$ $c_{i,j}$	obchod 1	obchod 2	obchod 3	
výrobce 1	3	55 2	2	5
výrobce 2	40 1	4	3	0
výrobce 3	5	2	4	55
	25	0	35	

Jelikož se nám nuluje sloupec s indexem $j = 2$ dostáváme tedy množiny $\mathcal{I} = \{1, 3\}$ a $\mathcal{J} = \{1, 3\}$. Další možné pole s nejnižší cenou je $x_{1,3}$, které obsadíme hodnotou $x_{1,3} = 5$. Přepočítáme hodnoty $a_1 = 0$ a $b_3 = 30$.

$x_{i,j}$ $c_{i,j}$	obchod 1	obchod 2	obchod 3	
výrobce 1	3	55 2	5 2	0
výrobce 2	40 1	4	3	0
výrobce 3	5	2	4	55
	25	0	30	

Takto postupujeme dále až dostaneme řešení, které vypadá následovně

$x_{i,j}$ $c_{i,j}$	obchod 1	obchod 2	obchod 3	
výrobce 1	3	55 2	5 2	60
výrobce 2	40 1	4	3	40
výrobce 3	25 5	2	30 4	55
	65	55	35	

kde celková cena je $2 \cdot 55 + 2 \cdot 5 + 1 \cdot 40 + 5 \cdot 25 + 4 \cdot 30 = 330$. Můžeme vidět, že dostáváme počáteční řešení s menšími přepravními náklady než při užití metody severozápadního rohu.

2.2.3 Vogelova aproximační metoda (VAM)

Tato metoda je náročnější na výpočet ale obvykle dává řešení s menšími přepravními náklady než předchozí metody. Pro každý řádek a každý sloupec vypočítáme rozdíl (diferenci) mezi dvěma nejmenšími cenami. Vybereme řádek nebo sloupec s největším rozdílem a v něm políčko s nejmenší cenou. Bude-li největší diference stejná u více řádků či sloupců, budeme obsazovat pole v těchto řádcích nebo sloupcích s nejmenší cenou. Doplníme $x_{i,j} = \min\{a_i, b_j\}$. Opět přepočítáme a_i a b_j a vynecháme řádek nebo sloupec, kde se hodnota nuluje. A celý postup opakujeme.

Příklad. Pro náš příklad 2.2 vypočítáme diference a zapíšeme si je do tabulky.

$x_{i,j}$ $c_{i,j}$		obchod 1	obchod 2	obchod 3	
	Δ	Δ=2	Δ=0	Δ=1	
výrobce 1	Δ=0	3	2	2	60
výrobce 2	Δ=2	1	4	3	40
výrobce 3	Δ=2	5	2	4	55
		65	55	35	

Jak vidíme hodnota diference pro první sloupec a pro druhý a třetí řádek je stejná a největší. Proto hledáme v těchto dvou řádcích a ve sloupci pole, kde je cena nejnižší. To platí pro pole $x_{2,1}$ a to obsadíme hodnotou $x_{2,1} = \min\{a_2, b_1\} = \min\{40, 65\} = 40$. Přepočteme $a_2 = 0$ a $b_1 = 25$, vynecháme tedy druhý řádek a přepočítáme diference.

$x_{i,j}$ $c_{i,j}$		obchod 1	obchod 2	obchod 3	
	Δ	Δ=2	Δ=0	Δ=2	
výrobce 1	Δ=0	3	2	2	60
výrobce 2	-	40 1	4	3	0
výrobce 3	Δ=2	5	2	4	55
		25	55	35	

Opět nám vycházejí stejné hodnoty nejvyšších rozdílů. Pole s nejmenší cenou v těchto sloupcích a řádku je $x_{1,3}$ a to obsadíme hodnotou $x_{1,3} = 35$. Přepočteme $a_1 = 25$ a $b_3 = 0$, vynecháme tedy třetí sloupec. Znovu přepočítáme diference.

$x_{i,j}$ $c_{i,j}$		obchod 1	obchod 2	obchod 3	
	Δ	Δ=2	Δ=0	-	
výrobce 1	Δ=1	3	2	35 2	25
výrobce 2	-	40 1	4	3	0
výrobce 3	Δ=3	5	2	4	55
		25	55	0	

Největší diference je pro třetí řádek, nejmenší cena v tomto řádku je pro pole $x_{3,2}$, obsadíme ho $x_{3,2} = 55$. Přepočítáme $a_3 = 0$ a $b_2 = 0$, vyškrtneme pouze jeden z indexů a to například sloupcový.

$x_{i,j}$ $c_{i,j}$		obchod 1	obchod 2	obchod 3	
	Δ	Δ=2	-	-	
výrobce 1	-	3	2	35 2	25
výrobce 2	-	40 1	4	3	0
výrobce 3	-	5	55 2	4	0
		25	0	0	

Osadíme tedy poslední pole $x_{1,1}$ hodnotou $x_{1,1} = 25$, tím získáme počáteční řešení ve tvaru:

$x_{i,j}$ $c_{i,j}$	obchod 1	obchod 2	obchod 3	
výrobce 1	25 3	2	35 2	60
výrobce 2	40 1	4	3	40
výrobce 3	5	55 2	4	55
	65	55	35	

Celková cena přepravy je $C = 3 \cdot 25 + 2 \cdot 35 + 1 \cdot 40 + 2 \cdot 55 = 295$. Můžeme tedy vidět, že náklady na dopravu jsou ještě o něco menší než při užití předchozích metod.

2.3 Nalezení optimálního řešení

Pro ověření optimality vycházejme z duality. Nejprve si napíšeme duální úlohu pro dopravní úlohu. Na straně 10 jsme si již připravili duální úlohu pro úlohu distribučního typu. Stačí v rovnici (1.8) položit $d_{i,j} = 1$ a $e_{i,j} = 1$. Jelikož máme v naší primární úloze podmínky pouze ve tvaru rovnosti, odpadá podmínka $u_i \leq 0$. Poté duální úloha k primární úloze (2.1) dopravního problému je ve tvaru

$$\text{maximalizovat} \quad \sum_{i=1}^{m} u_i a_i + \sum_{j=1}^{n} v_j b_j$$

$$\text{za podmínek} \quad c_{i,j} - u_i - v_j \geq 0 \qquad \text{pro } i = 1, ..., m; j = 1, ..., n$$

kde u_i jsou duální proměnné pro první omezení $\sum_{j=1}^{n} x_{i,j}{=}a_i$ a v_j jsou duální proměnné pro druhá omezení $\sum_{i=1}^{m} x_{i,j}{=}b_j$. Tyto duální proměnné mají význam Lagrangeových multiplikátorů. V tabulce je při výpočtu nazýváme řádková a sloupcová čísla. Postup výpočtu je následující:

Algoritmus 2.4. *Máme počáteční řešení.*

1. *Vypočteme řádková čísla u_i a sloupcová čísla v_j z rovností*

$$u_i = c^B_{i,j} - v_j \tag{2.2}$$
$$v_j = c^B_{i,j} - u_i \tag{2.3}$$

kde $c^B_{i,j}$ jsou ceny bázových proměnných (tj. ceny vyplněných políček tabulky). Protože máme nejvýše $m+n-1$ lineárně nezávislých rovnic a $m+n$ neznámých, můžeme zvolit jednu proměnnou jako parametr. Například zvolme $u_1 = 1$ a další čísla u_i a v_j postupně dopočítáváme užitím rovností 2.2 a 2.3.

2. *Pro nebázové proměnné dopočítáme hodnoty*

$$s_{i,j} = c^N_{i,j} - u_i - v_j$$

kde $c^N_{i,j}$ jsou ceny nebázových proměnných. Z těchto hodnot nalezneme nejmenší $s_{p,q} = min\{s_{i,j}\}$.

   (a) *Je-li $s_{p,q} \geq 0$ (tj. jsou splněny všechny podmínky $c_{i,j} - u_i - v_j \geq 0$), máme optimální řešení a výpočet ukončíme.*

   (b) *Je-li $s_{p,q} < 0$, proměnná $x_{p,q}$ vstoupí do báze. Hodnotu $x_{p,q}$ zatím označíme jako parametr δ a vepíšeme ji do tabulky. Dále v tabulce najdeme uzavřený okruh, což je n-úhelník, jehož vrcholy jsou pouze některá bázová pole (ne nutně všechna) a vychází se z pole $x_{p,q}$. Hledaný n-úhelník se skládá pouze z horizontálních a vertikálních čar. Ve vrcholech okruhu odečítáme nebo přičítáme hodnotu δ tak, aby stále byly zachovány rovnosti $\sum_{j=1}^{n} x_{i,j} = a_i$ a $\sum_{i=1}^{m} x_{i,j} = b_j$. Hodnotu parametru δ určíme z $\delta = min\left\{x^-_{i,j}\right\}$, kde $x^-_{i,j}$ jsou hodnoty, kde se δ odečítá. A tabulku upravíme dosazením vypočtené hodnoty δ.*

3. *Vrátíme se ke kroku 1. a celý postup opakujeme.*

Příklad. Pro příklad 2.2 jsme metodou severo-západního rohu našli počáteční řešení.

$x_{i,j}$ $c_{i,j}$	obchod 1	obchod 2	obchod 3	
výrobce 1	60 3	2	2	60
výrobce 2	5 1	35 4	3	40
výrobce 3	5	20 2	35 4	55
	65	55	35	

Nyní otestujeme optimalitu. Vypočítáme řádková a sloupcová čísla u_i a v_j. Zvolíme si $u_1 = 1$ a poté dále postupně vypočítáváme další.

$$u_1 = 1 \quad \rightarrow \quad v_1 = c_{1,1} - u_1 = 2 \quad \rightarrow \quad u_2 = c_{2,1} - v_1 = -1 \quad \rightarrow$$
$$v_2 = c_{2,2} - u_2 = 5 \quad \rightarrow \quad u_3 = c_{3,2} - v_2 = -3 \quad \rightarrow \quad v_3 = c_{3,3} - u_3 = 7$$

$x_{i,j}$ $c_{i,j}$		obchod 1	obchod 2	obchod 3	
	$u_i \backslash v_j$	$v_1 = 2$	$v_2 = 5$	$v_3 = 7$	
výrobce 1	$u_1 = 1$	60 ^{3}	^{2}	^{2}	60
výrobce 2	$u_2 = -1$	5 ^{1}	35 ^{4}	^{3}	40
výrobce 3	$u_3 = -3$	^{5}	20 ^{2}	35 ^{4}	55
		65	55	35	

Z hodnot u_i a v_j dopočítáme hodnoty $s_{i,j}$ pro prázdná pole tabulky

$$s_{1,2} = c_{1,2} - u_1 - v_2 = -4, \qquad s_{1,3} = -6, \qquad s_{2,3} = -3, \qquad s_{3,1} = 6,$$

nejmenší hodnota je $s_{1,3} = -6$, protože je tato hodnota záporná, řešení není optimální a do báze nám vstoupí $x_{1,3}$ s hodnotou rovnou parametru δ. Pak hledáme uzavřený okruh vycházející z $x_{1,3}$ a procházející bázovými prvky (obsazenými políčky).

$x_{i,j}$ $c_{i,j}$		obchod 1	obchod 2	obchod 3	
	$u_i \backslash v_j$	2	5	7	
výrobce 1	1	$60 - \delta$ ^{3}	^{2}	δ ^{2}	60
výrobce 2	-1	$5 + \delta$ ^{1}	$35 - \delta$ ^{4}	^{3}	40
výrobce 3	-3	^{5}	$20 + \delta$ ^{2}	$35 - \delta$ ^{4}	55
		65	55	35	

Z hodnot, kde se δ odečítá, je nejmenší hodnota 35 a tak $\delta = 35$. Nyní se nám nulují dva prvky, ale z báze může vypadnout pouze jeden prvek. Ponecháme si tedy pole $x_{3,3} = 0$ v bázi. Protože je prvek $x_{3,3} = 0$ bázový zapisujeme ho do tabulky. Takový prvek nazývejme bázová nula.

$x_{i,j}$ $c_{i,j}$	obchod 1	obchod 2	obchod 3	
výrobce 1	25 ^{3}	^{2}	35 ^{2}	60
výrobce 2	40 ^{1}	^{4}	^{3}	40
výrobce 3	^{5}	55 ^{2}	0 ^{4}	55
	65	55	35	

Znovu otestujeme optimalitu, nejprve vypočítáme u_i a v_j.

$x_{i,j}$ $c_{i,j}$		obchod 1	obchod 2	obchod 3	
	$u_i \diagdown v_j$	2	-1	1	
výrobce 1	1	25 ^{3}	^{2}	35 ^{2}	60
výrobce 2	-1	40 ^{1}	^{4}	^{3}	40
výrobce 3	3	^{5}	55 ^{2}	0 ^{4}	55
		65	55	35	

Poté vypočítáme $s_{i,j}$ pro nebázové hodnoty:

$$s_{1,2} = 0, \qquad s_{2,2} = 4, \qquad s_{2,3} = 3, \qquad s_{3,1} = 0$$

Minimální hodnota je $s_{1,2} = 0$ a $s_{3,1} = 0$, které jsou nezáporné. Nalezli jsme tedy optimální řešení.

$x_{i,j}$ $c_{i,j}$	obchod 1	obchod 2	obchod 3	
výrobce 1	25 ^{3}	^{2}	35 ^{2}	60
výrobce 2	40 ^{1}	^{4}	^{3}	40
výrobce 3	^{5}	55 ^{2}	0 ^{4}	55
	65	55	35	

Toto řešení nám říká, že od 1. výrobce se přepraví 25 židlí do obchodu 1 a 35 židlí do obchodu 3, 40 židlí od 2. výrobce do 1. obchodu a 55 židlí od 3. výrobce do obchodu 2. Celkové přepravní náklady jsou $c = 3 \cdot 25 + 2 \cdot 35 + 1 \cdot 40 + 2 \cdot 55 = 295$.

Poznámka. Když se vrátíme zpět, všimneme si, že jsme optimální řešení dostali již Vogelovou aproximační metodou. Tento výsledek neplatí vždy, často nastává u malých úloh, ale ani tam to není zaručeno.

3 Distribuční úloha

Distribuční úloha je zobecněním dopravní úlohy. Popisuje problém, kde se jednotky dodavatelů a odběratelů liší. Jedná se převážně o úlohy optimalizace výrobního procesu. V dalším textu se budeme zabývat distribuční úlohou ve tvaru:

$$\begin{aligned}
&\text{minimum} && \sum_{i=1}^{m}\sum_{j=1}^{n} c_{i,j}x_{i,j} && \\
&\text{za podmínek} && \sum_{j=1}^{n} x_{i,j} \le a_i && i = 1,2,...,m \\
& && \sum_{i=1}^{m} d_{i,j}x_{i,j} = b_j && j = 1,2,...,n \\
& && x_{i,j} \ge 0 && i = 1,2,...,m; j = 1,2...,n
\end{aligned} \tag{3.1}$$

Koeficienty $a_i \ge 0$ jsou kapacity, koeficienty $b_j \ge 0$ jsou požadovaná množství, $c_{i,j} \ge 0$ jsou cenové koeficienty a $d_{i,j}$ jsou koeficienty výkonnosti. Tyto koeficienty nám udávají vztah mezi jednotkami koeficientů a_i a b_j. Obecně mohou být libovolné, my budeme brát pouze $d_{i,j} \ge 0$. Při zápornosti koeficientů $d_{i,j}$ by úloha mohla být neomezená a navíc nemají praktický význam. Později si ukážeme, že úloha (3.1) je ekvivalentní s obecnější úlohou (1.7) uvedenou na straně 10.

Distribuční úlohu lze obecně zapsat do tabulky následovně

$d_{1,1}\ x_{1,1}\ c_{1,1}$	$d_{1,2}\ x_{1,2}\ c_{1,2}$	...	$d_{1,n}\ x_{1,n}\ c_{1,n}$	a_1
$d_{2,1}\ x_{2,1}\ c_{2,1}$	$d_{2,2}\ x_{2,2}\ c_{2,2}$	...	$d_{2,n}\ x_{2,n}\ c_{2,n}$	a_2
$\vdots$	$\vdots$	$\ddots$	$\vdots$	
$d_{m,1}\ x_{m,1}\ c_{m,1}$	$d_{m,2}\ x_{m,2}\ c_{m,2}$	...	$d_{m,n}\ x_{m,n}\ c_{m,n}$	a_m
b_1	b_2		b_n	

V této tabulce také provádíme výpočet řešení.

Převod na rovnicový tvar

V úloze máme obecně dané podmínky u řádkových součtů ve tvaru nerovnosti. Při výpočtu optimálního řešení budeme předpokládat, že máme podmínky ve tvaru rovnosti. Proto musíme přidat další proměnné $x_{i,n+1}$, v tabulce to odpovídá dalšímu přidanému $(n+1)$. sloupci. Koeficienty výkonnosti $d_{i,j}$ u těchto proměnných budou rovny $d_{i,n+1} = 1$ a ceny $c_{i,n+1} = 0$ pro $i = 1,2,...,m$. Tyto proměnné nazveme **doplňkové**. Z

úlohy (3.1) dostaneme úlohu v rovnicovém tvaru

$$\begin{aligned}
&\text{minimalizovat } \sum_{i=1}^{m}\sum_{j=1}^{n} c_{i,j}x_{i,j} \\
&\text{za podmínek } \sum_{j=1}^{n+1} x_{i,j} = a_i \qquad i = 1,2,...,m \\
&\qquad\qquad \sum_{i=1}^{m} d_{i,j}x_{i,j} = b_j \qquad j = 1,2,...,n \\
&\qquad\qquad x_{i,j} \geq 0 \qquad i = 1,2,...,m; j = 1,2...,n+1
\end{aligned} \tag{3.2}$$

Poznámka. V úloze v rovnicovém tvaru (3.2) si povšimněme, že pro doplňkové proměnné není omezení $\sum_{i=1}^{m} d_{i,n+1}x_{i,n+1} = b_{n+1}$.

Pokud nalezneme optimální řešení, kde je hodnota doplňkové proměnné $x_{i,n+1} \geq 0$ pro nějaké $i = 1,2,...,m$, tak toto řešení je optimální řešení i původní úlohy ve tvaru (3.1). Kladná hodnota $x_{i,n+1}$ značí nevyužitou kapacitu i-tého stroje.

Úlohu v této podobě si můžeme zapsat i v maticovém tvaru

$$\begin{aligned}
&\text{minimalizovat } \mathbf{c^T x} \\
&\text{za podmínky } \mathbf{Ax} = \mathbf{b} \\
&\qquad\qquad \mathbf{x} \geq 0
\end{aligned}$$

kde

$$\begin{aligned}
&\text{vektor } \mathbf{c} = (c_{1,1}, c_{1,2}, ..., c_{1,n+1}, c_{2,1}, ..., c_{m,n+1})^T \\
&\text{vektor } \mathbf{x} = (x_{1,1}, x_{1,2}, ..., x_{1,n+1}, x_{2,1}, ..., x_{m,n+1})^T \\
&\text{vektor } \mathbf{b} = (a_1, a_2, ..., a_m, b_1, b_2, ..., b_n)^T \\
&\text{matice } \mathbf{A} \text{ má následující tvar}
\end{aligned}$$

$$\mathbf{A} = \left(\begin{array}{ccccc|ccccc|c|ccccc}
1 & 1 & \cdots & 1 & 1 & & & & & & & & & & & \\
 & & & & & 1 & 1 & \cdots & 1 & 1 & & & & & & \\
 & & & & & & & & & & \ddots & & & & & \\
 & & & & & & & & & & & 1 & 1 & \cdots & 1 & 1 \\
\hline
d_{1,1} & & & & & d_{2,1} & & & & & & d_{m,1} & & & & \\
 & d_{1,2} & & & & & d_{2,2} & & & & \cdots & & d_{m,2} & & & \\
 & & \ddots & & & & & \ddots & & & & & & \ddots & & \\
 & & & d_{1,n} & 0 & & & & d_{2,n} & 0 & & & & & d_{m,n} & 0
\end{array}\right)$$

Věta 3.1. *Hodnost matice* **A** *je rovna* $m + n$.

Důkaz. Vezmeme-li postupně poslední sloupce z každého bloku a následně postupně prvních n sloupců z libovolného bloku. Tímto dostaneme horní trojúhelníkovou matici s $m + n$ nezávislými sloupci. Jelikož počet nezávislých sloupců je stejný jako počet nezávislých řádků, je hodnost matice právě $m + n$. □

3.1 Příklady aplikací

Následující příklady nám pomohou lépe pochopit význam všech proměnných a koeficientů. Nejprve si ukážeme obecné příklady aplikací.

Nejčastější aplikace distribuční úlohy je právě úloha optimalizace výrobního procesu.

Úloha. Podnik má nastavit výrobu produktů na výrobních strojích tak, aby zároveň minimalizoval výrobní náklady. Maximální počet hodin, které může i-tý stroj pracovat za dané období označme a_i. Množství j-tého produktu, které je třeba vyrobit označme b_j. Koeficienty $d_{i,j}$ určují, kolik j-tého produktu vyrobí i-tý stroj za hodinu. Ceny výroby j-tého produktu i-tým strojem označme $c_{i,j}$. Hledáme optimální rozvržení výroby, tedy počet hodin, ve kterých bude i-tý stroj vyrábět j-tý produkt (ozn. $x_{i,j}$).

Úloha. Eskadra složená z různých typů letadel se má nasadit na letecké tratě tak, aby uspokojila poptávku cestujících a zároveň minimalizovat přepravní náklady. Počet letadel i-tého typu označme a_i, počet cestujících požadující přepravu na trati j označme b_j, koeficienty $d_{i,j}$ nám určují počet cestujících, které je možno přepravit jedním letadlem i-tého typu, pokud je nasazené na j-tou leteckou trať během daného období. Přepravní náklady na jedno letadlo i-tého typu na j-té letecké trati. Hledáme tedy rozvržení letů, to je počet letadel i-tého typu na j-té letecké trati označme $x_{i,j}$. Viz [6, str.477]

Úloha. Podnik vyrábí energii. Má k dispozici a_i množství i-té suroviny pro výrobu energie. Potřebné množství vyrobené energie v j-tém zařízení označme b_j. Koeficienty $d_{i,j}$ označují spotřebu i-té suroviny v j-tém zařízení (zahrnují například potřebnost konverze suroviny apod.). Označme $p_{i,j}$ náklady na výrobu jednotkového množství energie z i-té suroviny v j-tém zařízení. Hledáme spotřebu i-té suroviny v j-tém zařízení (ozn. $x_{i,j}$) tak, abychom minimalizovali náklady $c_{i,j} = d_{i,j}p_{i,j}$ na výrobu energie v j-tém zařízení z jednotkového množství i-té suroviny. Viz [9].

Další příklady distribuční úlohy můžeme najít v [10, str. 70-72] nebo v [5, kap. 15], kde jsou příklady i řešeny. Nyní si uvedeme jednoduché příklady již s konkrétními čísly, na kterých budeme demonstrovat metody pro výpočet řešení.

Příklad 3.2. Firma vyrábí plastové hračky - 4 druhy zvířátek. Pro tento účel má k dispozici tři linky - vstřikovací lisy. Požadavky odběratelů (ozn. b_j) jsou 220 ks psů, 150 ks koček, 180 ks slonů a 100 ks žiraf. Měsíční kapacita linek (a_i) je 160 hod., 320 hod. a 200 hod. pro jednotlivé linky. Následující tabulka zobrazuje, kolik jednotlivých zvířátek (v ks) vyrobí daná linka za hodinu ($d_{i,j}$) a cenu hodiny provozu ($c_{i,j}$) (v tis. Kč) dané linky při výrobě daného druhu zvířátka.

$d_{i,j}$ $c_{i,j}$	pes	kočka	slon	žirafa	
Linka A	0,5 7	0,8 6	0,5 5	0,7 4	160
Linka B	0,7 6	0,4 8	0,6 5	0,5 3	320
Linka C	0,4 4	0,6 5	0,6 6	0,8 5	200
	220	150	180	100	

Tato úloha je převzata z [8]. Jak si později ukážeme, nemá optimální řešení, což lze ověřit užitím jednoduché nutné podmínky

Příklad 3.3. Upravením předchozího příkladu zvýšením koeficientů výkonnosti $d_{i,j}$ dostaneme příklad ve tvaru zapsaném v následující tabulce.

$d_{i,j}$ $c_{i,j}$	pes	kočka	slon	žirafa	
Linka A	0,6 7	1 6	0,6 5	0,8 4	160
Linka B	0,8 6	0,5 8	0,7 5	0,6 3	320
Linka C	0,5 4	0,7 5	0,7 6	1 5	200
	220	150	180	100	

Jak si později ukážeme i tento příklad nemá optimální řešení, přestože splňuje jednoduchou nutnou podmínku existence optimálního řešení. Později to ověříme výpočtem.

Abychom dostali příklad, který má optimální řešení a mohli na něm ukázat postupy pro nalezení řešení, musíme opět upravit koeficienty výkonnosti $d_{i,j}$ a také kapacity linek a_i. Po upravení dostaneme následující příklad, se kterým budeme dále pracovat.

Příklad 3.4. Firma vyrábí plastové hračky – 4 druhy zvířátek. Pro tento účel má k dispozici tři linky – vstřikovací lisy. Požadavky odběratelů jsou 220 ks psů, 150 ks koček, 180 ks slonů a 100 ks žiraf. Měsíční kapacita linek je 240 hod., 320 hod. a 220 hod. pro jednotlivé linky. Následující tabulka zobrazuje, kolik jednotlivých zvířátek (v ks) vyrobí daná linka za hodinu a cenu hodiny provozu (v tis. Kč) dané linky při výrobě daného druhu zvířátka. Hledáme optimální rozvržení výroby nebo-li počet hodin práce ($x_{i,j}$) dané linky na daném výrobku tak, aby náklady na výrobu byly minimální.

$d_{i,j}$ $c_{i,j}$	pes	kočka	slon	žirafa	
Linka A	0,8 7	1,2 6	0,8 5	1 4	240
Linka B	1 6	0,5 8	0,9 5	0,8 3	320
Linka C	0,6 4	0,8 5	0,9 6	1,2 5	220
	220	150	180	100	

S tímto příkladem budeme dále pracovat a ukážeme si na něm postupy pro nalezení řešení.

Můžeme jej zapsat i v maticovém tvaru

$$\begin{aligned} &\text{minimalizovat } \mathbf{c^T x} \\ &\text{za podmínky } \mathbf{Ax = b} \\ &\mathbf{x} \geq 0 \end{aligned}$$

kde

$$\mathbf{c} = \begin{pmatrix} 7, & 6, & 5, & 4, & 0, & 6, & 8, & 5, & 3, & 0, & 4, & 5, & 6, & 5, & 0 \end{pmatrix}^T$$

$$\mathbf{x} = (x_{1,1},\, x_{1,2},\, x_{1,3},\, x_{1,4},\, x_{1,5},\, x_{2,1},\, x_{2,2},\, x_{2,3},\, x_{2,4},\, x_{2,5},\, x_{3,1},\, x_{3,2},\, x_{3,3},\, x_{3,4},\, x_{3,5})^T$$

$$\mathbf{A} = \left(\begin{array}{ccccc|ccccc|ccccc} 1 & 1 & 1 & 1 & 1 & & & & & & & & & & \\ & & & & & 1 & 1 & 1 & 1 & 1 & & & & & \\ & & & & & & & & & & 1 & 1 & 1 & 1 & 1 \\ - & - & - & - & - & - & - & - & - & - & - & - & - & - & - \\ 0,8 & & & & & 1 & & & & & 0,6 & & & & \\ & 1,2 & & & & & 0,5 & & & & & 0,8 & & & \\ & & 0,8 & & & & & 0,9 & & & & & 0,9 & & \\ & & & 1 & 0 & & & & 0,8 & 0 & & & & 1,2 & 0 \end{array}\right)$$

$$\mathbf{b} = \begin{pmatrix} 240, & 320, & 220, & 220, & 150, & 180, & 100 \end{pmatrix}$$

3.2 Ekvivalentní tvary distribuční úlohy

Distribuční úloha je zobecněním dopravní úlohy. Dalším zobecněním vznikne obecná distribuční úloha

$$\begin{aligned}
\text{minimalizovat} \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{i,j}x_{i,j} & & \\
\text{za podmínek} \quad & \sum_{j=1}^{n} e_{i,j}x_{i,j} \leq a_i & & i = 1,2,...,m \\
& \sum_{i=1}^{m} d_{i,j}x_{i,j} = b_j & & j = 1,2,...,n \\
& x_{i,j} \geq 0 & & i = 1,2,...,m;\ j = 1,2,...,m
\end{aligned} \tag{3.3}$$

Ukážeme si, že tato úloha je ekvivalentní s distribuční úlohou (3.1) a můžeme se tedy zabývat pouze úlohou (3.1). Převod úlohy (3.3) na úlohu (3.1), je naznačený v [6]. Abychom úlohu mohli převést, musí koeficienty splňovat podmínky $e_{i,j} > 0$, $d_{i,j} \geq 0$ a $c_{i,j} \geq 0$. Poté můžeme zavést substituci ve tvaru $\overline{x}_{i,j} = e_{i,j}x_{i,j}$, abychom z koeficientů u řádkových součtů dostali jedničky. Dosadíme ji do úlohy a dostáváme

$$\begin{aligned}
\text{minimalizovat} \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{i,j}\frac{\overline{x}_{i,j}}{e_{i,j}} & & \\
\text{za podmínek} \quad & \sum_{j=1}^{n} e_{i,j}\frac{\overline{x}_{i,j}}{e_{i,j}} \leq a_i & & i = 1,2,...,m \\
& \sum_{i=1}^{m} d_{i,j}\frac{\overline{x}_{i,j}}{e_{i,j}} = b_j & & j = 1,2,...,n \\
& \frac{\overline{x}_{i,j}}{e_{i,j}} \geq 0 & & i = 1,2,...,m;\ j = 1,2,...,m
\end{aligned}$$

Odtud již po upravení dostáváme konečnou podobu

$$\begin{aligned}
\text{minimalizovat} \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} \overline{c}_{i,j}\overline{x}_{i,j} & & \\
\text{za podmínek} \quad & \sum_{j=1}^{n} \overline{x}_{i,j} \leq a_i & & i = 1,2,...,m \\
& \sum_{i=1}^{m} \overline{d}_{i,j}\overline{x}_{i,j} = b_j & & j = 1,2,...,n \\
& \overline{x}_{i,j} \geq 0 & & i = 1,2,...,m;\ j = 1,2,...,m
\end{aligned}$$

kde $\overline{c}_{i,j} = \frac{c_{i,j}}{e_{i,j}}$, $\overline{d}_{i,j} = \frac{d_{i,j}}{e_{i,j}}$.

Obdobně můžeme převést úlohu

$$\begin{aligned}
&\text{minimalizovat} && \sum_{i=1}^{m}\sum_{j=1}^{n} c_{i,j}x_{i,j} && \\
&\text{za podmínek} && \sum_{j=1}^{n} e_{i,j}x_{i,j} = a_i && i = 1,2,...,m \\
& && \sum_{i=1}^{m} d_{i,j}x_{i,j} \leq b_j && j = 1,2,...,n \\
& && x_{i,j} \geq 0 && i = 1,2,...,m;\ j = 1,2,...,m
\end{aligned} \tag{3.4}$$

ve které opět platí $e_{i,j} \geq 0$, $d_{i,j} > 0$ a $c_{i,j} \geq 0$. Zvolíme substituci $\overline{x}_{i,j} = d_{i,j}x_{i,j}$ a konečnou podobu úlohy dostaneme ve tvaru

$$\begin{aligned}
&\text{minimalizovat} && \sum_{i=1}^{m}\sum_{j=1}^{n} \overline{c}_{i,j}\overline{x}_{i,j} && \\
&\text{za podmínek} && \sum_{j=1}^{n} \overline{e}_{i,j}\overline{x}_{i,j} = a_i && i = 1,2,...,m \\
& && \sum_{i=1}^{m} \overline{x}_{i,j} \leq b_j && j = 1,2,...,n \\
& && \overline{x}_{i,j} \geq 0 && i = 1,2,...,m;\ j = 1,2,...,m
\end{aligned}$$

kde $\overline{c}_{i,j} = \frac{c_{i,j}}{d_{i,j}}$, $\overline{e}_{i,j} = \frac{e_{i,j}}{d_{i,j}}$.

Není tedy nutno se zvláště zabývat distribuční úlohou ve tvaru (3.3) nebo (3.4), neboť obě úlohy lze převést na úlohu (3.1).

3.3 Vztah mezi dopravní a distribuční úlohou

Již dříve jsme se zmínili, že dopravní úloha je speciálním případem distribuční úlohy. Za určitých podmínek lze distribuční úlohu převést na dopravní úlohu (viz [2]), jejíž řešení je snadnější. Pokud existují kladná čísla α_i a β_j taková, že všechny koeficienty výkonnosti $d_{i,j}$ lze zapsat jako $d_{i,j} = \alpha_i\beta_j$, pak se dá distribuční úloha převést na úlohu

dopravní. Máme tedy distribuční úlohu ve tvaru

$$\begin{aligned}
\text{minimalizovat} \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{i,j}x_{i,j} & \\
\text{za podmínek} \quad & \sum_{j=1}^{n} x_{i,j} \leq a_i & i = 1,2,...,m \\
& \sum_{i=1}^{m} \alpha_i\beta_j x_{i,j} = b_j & j = 1,2,...,n \\
& x_{i,j} \geq 0 & i = 1,2,...,m; j = 1,2...,n
\end{aligned}$$

Použitím substituce $\overline{x}_{i,j} = \alpha_i x_{i,j}$, pak dostáváme

$$\begin{aligned}
\text{minimalizovat} \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{i,j}\frac{\overline{x}_{i,j}}{\alpha_i} & \\
\text{za podmínek} \quad & \sum_{j=1}^{n} \frac{\overline{x}_{i,j}}{\alpha_i} \leq a_i & i = 1,2,...,m \\
& \sum_{i=1}^{m} \alpha_i\beta_j \frac{\overline{x}_{i,j}}{\alpha_i} = b_j & j = 1,2,...,n \\
& \frac{\overline{x}_{i,j}}{\alpha_i} \geq 0 & i = 1,2,...,m; j = 1,2...,n
\end{aligned}$$

Úlohu upravíme na výsledný tvar

$$\begin{aligned}
\text{minimalizovat} \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} \overline{c}_{i,j}\overline{x}_{i,j} & \\
\text{za podmínek} \quad & \sum_{j=1}^{n} \overline{x}_{i,j} \leq \overline{a}_i & i = 1,2,...,m \\
& \sum_{i=1}^{m} \overline{x}_{i,j} = \overline{b}_j & j = 1,2,...,n \\
& \overline{x}_{i,j} \geq 0 & i = 1,2,...,m; j = 1,2...,n
\end{aligned}$$

kde $\overline{c}_{i,j} = \frac{c_{i,j}}{\alpha_i}$, $\overline{a}_{i,j} = \alpha_i a_{i,j}$, $b_j = \frac{b_j}{\beta_j}$. Problém je v nalezení čísel α_i a β_j, tato čísla často ani neexistují. Proto nelze libovolnou distribuční úlohu převést na dopravní úlohu a je nutné ji řešit speciálními metodami.

3.4 Řešitelnost úloh

Pro distribuční úlohu neexistuje žádné jednoduché pravidlo, které by nám zaručovalo, že má úloha optimální nebo alespoň přípustné řešení, jako jsme měli u dopravní úlohy.

Pro dopravní úlohu nám platí věta 2.1 na straně 12, kde nám stačí ověřit zda platí dodavatelsko-odběratelská rovnováha. Žádná taková podmínka u distribuční úlohy neplatí. Pro distribuční úlohu lze vyslovit pouze následující větu.

Věta 3.5. *Pro distribuční úloha* (3.1) *jsou následující tvrzení ekvivalentní:*

a)má přípustné řešení

b)má optimální řešení

Důkaz. $a) \Rightarrow b)$: Věta 1.6 o existenci optimálního řešení nám říká, že pokud máme přípustné řešení a účelová funkce je omezená zdola, pak existuje optimální řešení. Jelikož máme omezení $c_{i,j} \geq 0$ a $x_{i,j} \geq 0$, pak i účelová funkce je omezená zdola a tvrzení platí. $b) \Rightarrow a)$: Tvrzení plyne rovnou z definice optimálního řešení. □

Poznámka. Pokud aplikujeme základní větu lineárního programování 1.7, tak oproti obecné úloze lineárního programování pro distribuční úlohu nemůže nastat 2. možnost této věty, jelikož účelová funkce je omezená zdola. Z tohoto vyplývá, že buď neexistuje přípustné řešení nebo existuje optimální řešení distribuční úlohy.

Pro distribuční úlohu si můžeme zformulovat alespoň nutnou podmínku existence optimálního řešení ve tvaru

$$\max\{d_{i,j}\} \sum_{i=1}^{m} a_i \geq \sum_{j=1}^{n} b_j \tag{3.5}$$

Vyjdeme z podmínek

$$\sum_{j=1}^{n} x_{i,j} \leq a_i \qquad i = 1, 2, ..., m$$

$$\sum_{i=1}^{m} d_{i,j} x_{i,j} = b_j \qquad j = 1, 2, ..., n$$

Pokud $d_{i,j}$ nahradíme maximem z těchto hodnot, které si označme $d_{max} = \max\{d_{i,j}\}$, dostáváme

$$\sum_{j=1}^{n} x_{i,j} \leq a_i \qquad i = 1, 2, ..., m$$

$$\sum_{i=1}^{m} d_{max} x_{i,j} \geq b_j \qquad j = 1, 2, ..., n$$

Jestliže všechny tyto rovnice sečteme, dostáváme

$$\sum_{i=1}^{m}\sum_{j=1}^{n} x_{i,j} \leq \sum_{i=1}^{m} a_i$$
$$\sum_{i=1}^{m}\sum_{j=1}^{n} x_{i,j} \geq \frac{1}{d_{max}} \sum_{j=1}^{n} b_j$$

Z tohoto již dostáváme nutnou podmínku (3.5). Pokud tato podmínka bude porušena, nelze splnit podmínky úlohy (3.1) a úloha nebude mít ani přípustné řešení. Ovšem pokud podmínka splněna bude, nezaručuje nám, že úloha má přípustné řešení.

Příklad. Jak jsme si již uvedli, příklad 3.2 nemá optimální řešení. Ověřme si toto tvrzení podmínkou (3.5). Nejprve najdeme maximální hodnotu z koeficientů výkonnosti $\max\{d_{i,j}\} = 0,8$ a spočítejme $\sum_{i=1}^{m} a_i = 680$ a $\sum_{j=1}^{n} b_j = 650$. Po doplnění do nerovnosti dostáváme $544 \ngeq 650$. Je tedy porušena nutná podmínka, a proto tato úloha nemá řešení.

Příklad. Nyní si ukážeme na příkladu 3.3 tvrzení, že podmínka je splněna, ale nemusí existovat optimální ani přípustné řešení. Maximum z koeficientů výkonnosti je $\max\{d_{i,j}\} = 1$, součty omezení jsou stejné jako v předchozím příkladě, tedy $\sum_{i=1}^{m} a_i = 680$ a $\sum_{j=1}^{n} b_j = 650$. Po dosazení do (3.5) dostáváme $680 \geq 650$, tedy podmínka je splněna, dokonce platí ostrá nerovnost, ale přesto tato úloha nemá optimální řešení, což si můžeme ověřit například programem pro výpočet optimálního řešení, který je uvedený dále v textu.

Příklad. A poslední si uvedeme příklad 3.4, který má optimální řešení. Maximum z koeficientů výkonnosti je $\max\{d_{i,j}\} = 1,2$ a součty omezení $\sum_{i=1}^{m} a_i = 780$ a $\sum_{j=1}^{n} b_j = 650$. Po dosazení do (3.5) dostáváme $936 \geq 650$, a tedy je nutná podmínka splněna.

3.5 Nalezení počátečního bázového řešení

K nalezení výchozího řešení distribuční úlohy použijeme stejné či podobné metody jako pro nalezení výchozího řešení dopravní úlohy. Bohužel není žádný spolehlivý postup, který by nám dával přípustné počáteční řešení původní úlohy (3.1).

Doplnění pomocných proměnných

I po výpočtu počátečního řešení úlohy v rovnicovém tvaru dle níže uvedených postupů, tedy po vyplnění tabulky, se nám může stát, že stále nebudou splněny podmínky sloupcových součtů tedy $\sum_{i=1}^{m} d_{i,j} x_{i,j} < b_j \quad j \in \mathcal{K}$, kde $\mathcal{K} \subseteq \{1, 2, ..., n\}$ označme množinu

indexů, pro které neplatí rovnosti. Musíme tedy přidat další proměnné $x_{m+1,j}$ pro $j \in \mathcal{K}$. Nazveme je **pomocné proměnné**. Tyto proměnné nám ovšem změní úlohu. Výkonnostní koeficienty položíme opět rovny $d_{m+1,j} = 1$, ale ceny u těchto proměnných položíme rovny dostatečně velké kladné hodnotě $c_{m+1,j} = M$. Pokud dostaneme počáteční řešení s těmito proměnnými, musíme nejprve řešit úlohu

$$\begin{aligned}
\text{minimalizovat} \quad & \sum_{i=1}^{m}\sum_{j=1}^{n+1} c_{i,j}x_{i,j} + M\sum_{j\in\mathcal{K}} x_{m+1,j} & & \\
\text{za podmínek} \quad & \sum_{j=1}^{n+1} x_{i,j} = a_i & & i = 1,2,...,m \\
& \sum_{i=1}^{m} d_{i,j}x_{i,j} = b_j & & j \in \{1,2,...,n\} \setminus \mathcal{K} \\
& \sum_{i=1}^{m+1} d_{i,j}x_{i,j} = b_j & & j \in \mathcal{K} \\
& x_{i,j} \geq 0 & & i = 1,2,...,m;\ j = 1,2...,n+1 \\
& x_{i,j} \geq 0 & & i = m+1;\ j \in \mathcal{K}
\end{aligned} \tag{3.6}$$

Budeme tedy hledat optimální řešení této úlohy (viz kapitola 3.6) . Pokud během postupu dostaneme přípustné řešení, kde $x_{m+1,j} = 0$ pro všechny $j \in \mathcal{K}$, zbylé prvky nám tvoří přípustné řešení původní úlohy a plynule tak přejdeme k řešení původní úlohy. Proměnné $x_{m+1,j}$ již nebereme v úvahu. Nalezneme-li optimální řešení, kde se vyskytuje kladná hodnota $x_{m+1,j}$, není to optimální ani přípustné řešení původní úlohy a původní úloha řešení nemá.

V programu Matlab jsem pro přidání doplňkových (viz podkapitola převod na rovnicový tvar) a pomocných proměnných napsala funkci s názvem `rovnice`, která má hlavičku

```
function [x,c,d,iBx]=rovnice(x,c,d,na,nb,ha,hb)
```

Tuto funkci použijeme až po funkci, která nám vypočítá počáteční řešení některou níže uvedenou metodou. Tuto funkci zavoláme příkazem

```
[x,c,d,iBx]=rovnice(x,c,d,na,nb,ha,hb)
```

Všechny vstupní proměnné dostaneme z některé funkce pro výpočet počátečního řešení, které si uvedeme níže. Vystupující proměnné jsou matice po přidání doplňkových případně pomocných proměnných. A matice `iBx` je matice indikátorům bázových prvků v matici `x`. Pokud je prvek v `x` bázový, tak v matici `iBx` bude na tomto místě 1. Tuto matici zavádíme pro případ, kdyby se v řešení objevily tzv. bázové nuly. Tato funkce nejprve přidá doplňkové proměnné (převedení na rovnicový tvar) a poté je-li třeba

doplní pomocné proměnné. Funkce také upraví matici cen `c` a matici koeficientů výkonnosti `d`. Pokud bude počáteční řešení degenerované, to znamená, že počet kladných prvků bude menší než $m + n$ tedy `ha+hb`, pak se přidá do matice indikátorů bázových prvků další prvek.

Poznámka. V programu přidáme všechny pomocné proměnné (nejen pro $j \in \mathcal{K}$), ale zajistíme, že z báze mohou pouze vystupovat, nikoli do ní vstupovat.

3.5.1 Obecný algoritmus

Dál si ukážeme několik možných postupů pro nalezení počátečního řešení. Nejprve si podobně jako u dopravní úlohy zformulujeme obecný algoritmus, který nám nedává jednoznačný postup při výpočtu. Výpočet provádíme v tabulce.

Algoritmus 3.6. *Máme indexové množiny* $\mathcal{I} = \{1, 2, ..., m\}$ *a* $\mathcal{J} = \{1, 2, ..., n\}$.

1. *Zvolme libovolně indexy* $i \in \mathcal{I}$ *a* $j \in \mathcal{J}$. *Pole s těmito indexy obsadíme hodnotou* $x_{i,j} = min\left\{a_i, \frac{b_j}{d_{i,j}}\right\}$ *a poté hodnoty* a_i *a* b_j *přepočítáme, tj.* $a_i = a_i - x_{i,j}$ *a* $b_j = b_j - d_{i,j}x_{i,j}$.

2. *Z indexových množin vyškrtneme index* i *nebo* j *a to ten, pro který se vynulují požadavky (tj.* $a_i = 0$ *nebo* $b_j = 0$*). V tabulce to znamená, že vyškrtneme příslušný řádek či sloupec. Vždy vynecháme pouze jeden z indexů. V případě, že se vynulují hodnoty pro oba indexy, vynecháme takový index z indexové množiny, která má alespoň dvě hodnoty, pokud je to možné. Tím dostáváme upravené indexové množiny a upravenou tabulku.*

3. *Vrátíme se ke kroku 1. a celý postup opakujeme, dokud nevyčerpáme všechny indexy* i *a* j.

Jako poslední je-li třeba obsadíme doplňkové a pomocné proměnné.

Dále si uvedeme speciální varianty obecného algoritmu, které přesněji předepisují způsob výběru indexů i a j.

3.5.2 Metoda severozápadního rohu

Prvky $x_{i,j}$ budeme obsazovat od levého horního rohu. Proto první obsazený prvek bude vždy $x_{1,1}$ podle vzorce $x_{i,j} = min\{a_i, \frac{b_j}{d_{i,j}}\}$. Dále upravíme hodnoty a_i a b_j, tj. $a_i = a_i - x_{i,j}$ a $b_j = b_j - d_{i,j}x_{i,j}$. Když hodnota a_i bude nulová pokračujeme prvkem, který leží o řádek níže, tj. $x_{i+1,j}$. Když se vynuluje hodnota b_j pokračujeme obsazením prvku $x_{i,j+1}$ stejným způsobem.

Příklad. Tento postup je nám znám z dopravní úlohy, a proto zde ukážeme jen výsledné počáteční řešení příkladu 3.4 ve tvaru:

$d_{i,j}$ $x_{i,j}$ $c_{i,j}$	pes	kočka	slon	žirafa	
Linka A	0,8 240 7	1,2 6	0,8 5	1 4	0
Linka B	1 28 6	0,5 292 8	0,9 5	0,8 3	0
Linka C	0,6 4	0,8 5 5	0,9 200 6	1,2 15 5	0
	0	0	0	82	

Jak můžeme vidět v posledním sloupci není splněna podmínka, $b_4 - \sum_{i=1}^{3} d_{i,4}x_{i,4} = 82 \neq 0$. Musíme tedy přidat pomocnou proměnnou $x_{4,4}$ s koeficientem $d_{4,4} = 1$ a cenou $c_{4,4} = M$. Hodnota pole bude rovna zbývajícímu množství tak, aby byla splněna podmínka.

$d_{i,j}$ $x_{i,j}$ $c_{i,j}$	pes	kočka	slon	žirafa	
Linka A	0,8 240 7	1,2 6	0,8 5	1 4	240
Linka B	1 28 6	0,5 292 8	0,9 5	0,8 3	320
Linka C	0,6 4	0,8 5 5	0,9 200 6	1,2 15 5	220
Pomocné proměnné	–	–	–	1 82 M	
	220	150	180	100	

Toto počáteční řešení není počáteční řešení naší úlohy ale úlohy upravené. Musíme tedy hledat řešení, kde bude hodnota $x_{4,4} = 0$, postupem, který si ukážeme později. Celkové náklady jsou zde $\sum_{i=1}^{4}\sum_{j=1}^{4} c_{i,j}x_{i,j}$=$5459 + 82M$.

Pro výpočet počátečního řešení metodou severozápadního rohu jsem v Matlabu napsala funkci `szr`, která má hlavičku

```
function [x,na,nb,ha,hb]=szr(a,b,c,d)
```

Na vstupu máme vektor řádkových omezení `a`, vektor sloupcových omezení `b`, matici cenových koeficientů `c` a matici koeficientů výkonnosti `d`. Všechny tyto proměnné získáme ze zadání úlohy. Na výstupu máme matici řešení `x`, upravený vektor řádkových omezení `na`, upravený vektor sloupcových omezení `nb`, dimenze vektorů `a` a `b` označené jako `ha` a `hb`. Abychom dostali přípustné řešení, musíme přidat doplňkové a pomocné proměnné. Musíme tedy ještě zavolat funkci `rovnice`, kterou máme uvedenou na straně 33. Funkce zavoláme příkazy

```
[x,na,nb,ha,hb]=szr(a,b,c,d)
[x,c,d,iBx]=rovnice(x,c,d,na,nb,ha,hb)
```

Příklad. Pro náš příklad 3.4 si zadefinujeme příklad

```
a=[240;320;220];
b=[220;150;180;100];
c=[7,6,5,4;6,8,5,3;4,5,6,5];
d=[0.8,1.2,0.8,1;1,0.5,0.9,0.8;0.6,0.8,0.9,1.2];
```

Zavoláním

```
[x,na,nb,ha,hb]=szr(a,b,c,d);
[x,c,d,iBx]=rovnice(x,c,d,na,nb,ha,hb)
```

dostáváme řešení

```
x =
   240     0     0     0     0
    28   292     0     0     0
     0     5   200    15     0
     0     0     0    82     0
```

Snadno ověříme, že toto řešení je stejné jako počáteční řešení při ručním výpočtu v tabulce.

3.5.3 Obdoba metody minimální ceny

Indexy prvků, které budeme obsazovat, můžeme vybírat stejně jako u dopravní úlohy porovnáním cen. Nejdříve vybereme prvky s nejmenší cenou a obsadíme jej hodnotou $x_{i,j} = min\{a_i, \frac{b_j}{d_{i,j}}\}$ a přepočítáme a_i a b_j, vynecháme sloupec nebo řádek podle toho která hodnota se nám vynuluje a postup opakujeme. Můžeme ovšem hledat i minimum z podílů $\frac{c_{i,j}}{d_{i,j}}$. Tento podíl má především ekonomický smysl, dostáváme tak například prvky s nejmenší cenou na výrobu jednoho produktu, viz [5]. Další postup je stejný, ale výsledky jsou různé.

Příklad. Metodu si opět ukážeme na příkladě 3.4. Nejprve použijeme metodu, kde hledáme minimální ceny $c_{i,j}$. Minimální cenu má pole $x_{2,4}$ a tak jej obsadíme hodnotou $x_{2,4} = \min\left\{a_2, \frac{b_4}{d_{2,4}}\right\} = 125$. Přepočítáme $a_2 = a_2 - x_{2,4} = 195$ a $b_4 = b_4 - x_{2,4}d_{2,4} = 0$

$d_{i,j}$ $x_{i,j}$ $c_{i,j}$	pes	kočka	slon	žirafa	
Linka A	0,8 7	1,2 6	0,8 5	1 4	240
Linka B	1 6	0,5 8	0,9 5	0,8 125 3	195
Linka C	0,6 4	0,8 5	0,9 6	1,2 5	220
	220	150	180	0	

Vynuluje se nám čtvrtý sloupec a tak jej již nebereme v úvahu a dále prohledáváme tabulku. Pole s nejmenší cenou je $x_{3,1}$ a obsadíme jej hodnotou $x_{3,1} = 220$. Přepočítáme $a_3 = 0$ a $b_1 = 88$.

$d_{i,j}$ $x_{i,j}$ $c_{i,j}$	pes	kočka	slon	žirafa	
Linka A	0,8 7	1,2 6	0,8 5	1 4	240
Linka B	1 6	0,5 8	0,9 5	0,8 125 3	195
Linka C	0,6 220 4	0,8 5	0,9 6	1,2 5	0
	88	150	180	0	

Dále pokračujeme podobným způsobem až dostaneme tabulku

$d_{i,j}$ $x_{i,j}$ $c_{i,j}$	pes	kočka	slon	žirafa	
Linka A	0,8 7	1,2 15 6	0,8 225 5	1 4	0
Linka B	1 88 6	0,5 107 8	0,9 5	0,8 125 3	0
Linka C	0,6 220 4	0,8 5	0,9 6	1,2 5	0
	0	78,5	0	0	

Sice jsme vyčerpali všechny možnosti, ale stále nemáme splněnou podmínku pro druhý sloupec. Musíme tedy přidat pomocnou proměnnou $x_{4,2}$ s koeficienty $d_{4,2} = 1$ a s cenami $c_{4,2} = M$

$d_{i,j}$ $x_{i,j}$ $c_{i,j}$	pes	kočka	slon	žirafa	
Linka A	0,8 7	1,2 15 6	0,8 225 5	1 4	240
Linka B	1 88 6	0,5 107 8	0,9 5	0,8 125 3	320
Linka C	0,6 220 4	0,8 5	0,9 6	1,2 5	220
Pomocné proměnné	–	1 78,5 M	–	–	
	220	150	180	100	

Celkové náklady činí $3854 + 78,5M$.

Příklad. A nyní vypočteme příklad 3.4 i metodou, kdy hledáme minimum z podílů $\frac{c_{i,j}}{d_{i,j}}$. Tyto hodnoty si může dopsat do tabulky do levé části každého pole, aby se nám lépe počítalo. Z této tabulky se již snadno vyhledá nejmenší hodnota podílu $\frac{c_{i,j}}{d_{i,j}}$, což je zde hodnota pro pole $x_{2,4}$. Toto pole obsadíme hodnotou $x_{2,4} = \min\{a_2; b_4/d_{2,4}\} = \min\{320; 100/0,8\} = 125$. Přepočítáme $a_2 = 195$ a $b_4 = 0$.

$\frac{c_{i,j}}{d_{i,j}}$ $d_{i,j}\, x_{i,j}\, c_{i,j}$	pes		kočka		slon		žirafa		
Linka A	8,75	0,8 7	5	1,2 6	6,25	0,8 5	4	1 4	240
Linka B	6	1 6	16	0,5 8	5,56	0,9 5	3,75	0,8 125 3	195
Linka C	6,67	0,6 4	6,25	0,8 5	6,67	0,9 6	4,17	1,2 5	220
	220		150		180		0		

Pokračujeme stejným postupem dále, dokud nedostaneme počáteční řešení v následujícím tvaru.

$d_{i,j}$ $x_{i,j}$ $c_{i,j}$	pes	kočka	slon	žirafa	
Linka A	0,8 $\frac{875}{8}$ 7	1,2 125 6	0,8 $\frac{45}{8}$ 5	1 4	240
Linka B	1 6	0,5 8	0,9 195 5	0,8 125 3	320
Linka C	0,6 220 4	0,8 5	0,9 6	1,2 5	220
Pomocné proměnné	1 $0,5$ M	–	–	–	
	220	150	180	100	

Celkové náklady jsou $\frac{30190}{8}+0,5M = 3773,75+0,5M$. Jak si můžeme všimnout výsledné řešení je odlišné od řešení, kdy jsme hledali minima pouze z cen $c_{i,j}$.

Pro výpočet počátečního řešení metou minimální ceny, kde hledáme minimum z cen $c_{i,j}$ nebo minimum z podílů $\frac{c_{i,j}}{d_{i,j}}$, jsem napsala funkci `mc` s hlavičkou

```
function [x,na,nb,ha,hb]=mc(a,b,c,d,varianta)
```

Na vstupu máme opět vektory a matice ze zadání a navíc proměnnou `varianta`, kterou si zvolíme variantu metody. Na výstupu máme matici řešení `x` a opravené vektory omezení `na` a `nb` a dimenze vektorů `a` a `b` označené `ha` a `hb`. V programu stejně jako v ručním postupu máme indexové množiny $\mathcal{I}$ a $\mathcal{J}$. Nejprve si zavedeme novou matici `nc` podle varianty metody, kterou jsme si zvolili. Prvky z matice, které již nebereme v úvahu, předefinují jako nekonečno a tedy tyto prvky nemohou být vybrány. Po výpočtu musíme opět zavolat funkci pro doplnění pomocných a doplňkových proměnných. Pro výpočet metodou, kde hledáme minimum z cen $c_{i,j}$ voláme

```
[x,na,nb,ha,hb]=mc(a,b,c,d,1);
[x,c,d,iBx]=rovnice(x,c,d,na,nb,ha,hb)
```

Pro výpočet počátečního řešení obdobou metody minimální ceny, kde hledáme minimum z podílů $\frac{c_{i,j}}{d_{i,j}}$ voláme funkci `mc`, kde jako pátý argument zadáme 0.

```
[x,na,nb,ha,hb]=mc(a,b,c,d,0);
[x,c,d,iBx]=rovnice(x,c,d,na,nb,ha,hb)
```

Příklad. Vektory a matice pro zadání příkladu 3.4 zadáme stejně jako jsme to udělali v příkladě pro metodu severozápadního rohu. Po zavolání funkcí

```
[x,na,nb,ha,hb]=mc(a,b,c,d,1);
[x,c,d,iBx]=rovnice(x,c,d,na,nb,ha,hb)
```

dostáváme řešení ve tvaru

```
x =
         0   15.0000  225.0000         0         0
   88.0000  107.0000         0  125.0000         0
  220.0000         0         0         0         0
         0   78.5000         0         0         0
```

Pokud budeme chtít hledat minima z podílu $\frac{c_{i,j}}{d_{i,j}}$, zavoláme funkce

```
[x,na,nb,ha,hb]=mc(a,b,c,d,0);
[x,c,d,iBx]=rovnice(x,c,d,na,nb,ha,hb)
```

a dostáváme řešení ve tvaru

```
x =
  109.3750  125.0000    5.6250         0         0
         0         0  195.0000  125.0000         0
  220.0000         0         0         0         0
    0.5000         0         0         0         0
```

3.5.4 Obdoba Vogelovy aproximační metody

Pro každý sloupec vypočítáme rozdíl dvou nejmenších podílů $\frac{c_{i,j}}{d_{i,j}}$, označme jej Δ. Poté vybereme sloupec s největším rozdílem a v tom pak obsadíme ten prvek, kde hodnota $\frac{c_{i,j}}{d_{i,j}}$ bude nejmenší, a to již známým vzorcem $x_{i,j} = min\{a_i, \frac{b_j}{d_{i,j}}\}$ a přepočítáme hodnoty a_i a b_j. Vynecháme příslušný řádek či sloupec a postup opakujeme.

Poznámka. U Vogelovy metody u dopravní úlohy jsme počítali diference jak pro sloupce tak pro řádky, zde počítáme pouze pro sloupce [5], je ovšem možné počítat diference i pro řádky.

Příklad. Opět si tento postup ukážeme na příkladě 3.4. Hodnoty podílů $\frac{c_{i,j}}{d_{i,j}}$ máme vypočteny již z předchozí metody, stačí nám dopočítat pouze rozdíly dvou nejmenších hodnot těchto podílů pro každý sloupec. Největší hodnota je pro druhý sloupec. V tomto sloupci je nejmenší hodnota podílu pro pole $x_{1,2}$ a toto pole obsadíme $x_{1,2} = \min\left\{a_1, \frac{b_2}{d_{1,2}}\right\} = \min\left\{240, \frac{150}{1,2}\right\} = 125$. Přepočítáme $a_1 = 240 - 125 = 115$ a $b_2 = 150 - 125 \cdot 1,2 = 0$.

$\frac{c_{i,j}}{d_{i,j}}$ $d_{i,j}x_{i,j}c_{i,j}$	pes		kočka		slon		žirafa		
Δ	$0,67$		$1,25$		$0,69$		$0,25$		
Linka A	8,75	0,8 7	5	1,2 125 6	6,25	0,8 5	4	1 4	115
Linka B	6	1 6	16	0,5 8	5,56	0,9 5	3,75	0,8 3	320
Linka C	6,67	0,6 4	6,25	0,8 5	6,67	0,9 6	4,17	1,2 5	220
	220		0		180		100		

Vynecháme druhý sloupec. Jelikož jsme vynechávali sloupec není potřeba přepočítávat rozdíly. Vybereme tedy další sloupec s největším rozdílem, tedy třetí sloupec a v něm pole s nejmenší hodnotou podílu je $x_{2,3}$, obsadíme ho $x_{2,3} = \min\{320; 180/0,9\} = 200$. Přepočítáme $a_2 = 120$ a $b_3 = 0$.

$\frac{c_{i,j}}{d_{i,j}}$ $d_{i,j}x_{i,j}c_{i,j}$	pes		kočka		slon		žirafa		
Δ	$0,67$		–		$0,69$		$0,25$		
Linka A	8,75	0,8 7	5	1,2 125 6	6,25	0,8 5	4	1 4	115
Linka B	6	1 6	16	0,5 8	5,56	0,9 200 5	3,75	0,8 3	120
Linka C	6,67	0,6 4	6,25	0,8 5	6,67	0,9 6	4,17	1,2 5	220
	220		0		0		100		

Opět se nám nuluje podmínka pro sloupec, a tak pouze vybereme další největší hodnotu ze zbylých rozdílů. To je pro první sloupec a v něm pole $x_{2,1}$, obsadíme ho hodnotou $x_{2,1} = \min\{120; 220/1\} = 120$. Přepočítáme $a_2 = 0$ a $b_1 = 100$.

$\frac{c_{i,j}}{d_{i,j}}$ $d_{i,j}x_{i,j}c_{i,j}$	pes		kočka		slon		žirafa		
Δ	$0,67$		–		–		$0,25$		
Linka A	8,75	0,8 7	5	1,2 125 6	6,25	0,8 5	4	1 4	115
Linka B	6	1 120 6	16	0,5 8	5,56	0,9 200 5	3,75	0,8 3	0
Linka C	6,67	0,6 4	6,25	0,8 5	6,67	0,9 6	4,17	1,2 5	220
	100		0		0		100		

Tentokrát se nám nuluje řádková podmínka, a tedy musíme přepočítat rozdíly. A největší rozdíl je opět pro první sloupec a nejmenší hodnota podílu je pro pole $x_{3,1}$ obsadíme ho $x_{3,1} = \min\{220; 100/0,6\} = \frac{500}{3}$. Přepočítáme $a_3 = \frac{160}{3}$ a $b_1 = 0$.

$\frac{c_{i,j}}{d_{i,j}}$ $d_{i,j}x_{i,j}c_{i,j}$	pes		kočka		slon		žirafa		
Δ	$2,08$		–		–		$0,25$		
Linka A	8,75	0,8 7	5	1,2 125 6	6,25	0,8 5	4	1 4	115
Linka B	6	1 120 6	16	0,5 8	5,56	0,9 200 5	3,75	0,8 3	0
Linka C	6,67	0,6 $\frac{500}{3}$ 4	6,25	0,8 5	6,67	0,9 6	4,17	1,2 5	$\frac{160}{3}$
	0		0		0		100		

Nyní se nám nuluje sloupcová podmínka, zbývá nám tedy už jen poslední sloupec. Vybereme zde pole s nejmenším možným podílem, proto obsadíme pole $x_{1,4}$ hodnotou $x_{1,4} = \min\{115; 100/1\} = 100$. Přepočítáme $a_1 = 115-100 = 15$ a $b_4 = 100-1\cdot 100 = 0$.

$\frac{c_{i,j}}{d_{i,j}}$ $d_{i,j}x_{i,j}{}^{c_{i,j}}$	pes		kočka		slon		žirafa		
Δ	–		–		–		0,25		
Linka A	8,75	0,8 7	5	1,2 125 6	6,25	0,8 5	4	1 100 4	15
Linka B	6	1 120 6	16	0,5 8	5,56	0,9 200 5	3,75	0,8 3	0
Linka C	6,67	0,6 $\frac{500}{3}$ 4	6,25	0,8 5	6,67	0,9 6	4,17	1,2 5	$\frac{160}{3}$
	0		0		0		0		

Vynecháme tedy poslední sloupec. Tím máme splněny sloupcové podmínky, ale řádkové podmínky pro první a třetí řádek nejsou splněny, přidáme proto sloupec s doplňkovými proměnnými a tím dostáváme počáteční řešení.

$d_{i,j}x_{i,j}{}^{c_{i,j}}$	pes	kočka	slon	žirafa	doplňkový	
Linka A	0,8 7	1,2 125 6	0,8 5	1 100 4	1 15 0	240
Linka B	1 120 6	0,5 8	0,9 200 5	0,8 3	1 0	320
Linka C	0,6 $\frac{500}{3}$ 4	0,8 5	0,9 6	1,2 5	1 $\frac{160}{3}$ 0	220
	220	150	180	100		

V tomto počátečním řešení se nám již neobjevují pomocné proměnné, máme tedy počáteční řešení naší původní úlohy. Celkové náklady činí $\frac{10610}{3} = 3536,\overline{66}$.

Pro Vogelovu aproximační metodu v programu Matlab jsem napsala funkci `VAM`, která má hlavičku

```
function [x,na,nb,ha,hb]=VAM(a,b,c,d,varianta,postup)
```

V této funkci si můžeme zvolit variantu této metody. Většina vstupních a výstupních proměnných je stejná jako v předchozích metodách. Vysvětlíme si tedy pouze proměnné `varianta` a `postup`. Proměnnou `varianta` si určíme, zda budeme počítat diference z cen `c` po zadání 1 nebo z podílů `c/d` po zadání 0. Proměnná `postup` určuje, zda chceme v metodě počítat diference pouze u sloupců po zadání 1 nebo u sloupců i řádků po zadání 0. Tímto si můžeme zvolit, jakým způsobem se bude počáteční řešení počítat. Program funguje na stejném principu jako metoda minimální ceny, takže prvky, které nebereme v úvahu, se položí rovny nekonečnu. Poté opět musíme použít funkci pro přidání pomocných a doplňkových proměnných. Funkce zavoláme

```
[x,na,nb,ha,hb]=VAM(a,b,c,d,varianta,postup);
[x,c,d,iBx]=rovnice(x,c,d,na,nb,ha,hb)
```

Příklad. Matice a vektory zadání příkladu 3.4 zadáme stejně jako u metod výše. V ručním výpočtu jsem počítali diference z podílů $\frac{c_{i,j}}{d_{i,j}}$ a to pouze pro sloupce. Abychom dostali stejné řešení jako při ručním výpočtu a mohli je tak srovnat, zadáme

```
[x,na,nb,ha,hb]=VAM(a,b,c,d,0,1);
[x,c,d,iBx]=rovnice(x,c,d,na,nb,ha,hb)
```

a dostáváme řešení

```
x =
         0   125.0000         0   100.0000    15.0000
  120.0000         0   200.0000         0         0
  166.6667         0         0         0    53.3333
```

které je stejné jako při ručním výpočtu.

Pokud budeme počítat příklad, kde budeme hledat diference z podílů $\frac{c_{i,j}}{d_{i,j}}$ pro sloupce i řádky, zavoláme funkci `VAM`, kde šestý parametr položíme rovno 0.

```
[x,na,nb,ha,hb]=VAM(a,b,c,d,0,0);
[x,c,d,iBx]=rovnice(x,c,d,na,nb,ha,hb)
```

Poté dostáváme řešení

```
x =
         0   125.0000   115.0000         0         0
  220.0000         0    97.7778         0    2.2222
         0         0         0   83.3333  136.6667
```

Povšimneme si, že řešení je odlišné od řešení, které je nalezené metodou, kde jsme hledali pouze diference pro sloupce.

Pro výpočet příkladu metodou, kde počítáme rozdíly z cen $c_{i,j}$ a to pouze pro sloupce zadáme

```
[x,na,nb,ha,hb]=VAM(a,b,c,d,1,1);
[x,c,d,iBx]=rovnice(x,c,d,na,nb,ha,hb)
```

a dostáváme řešení

```
x =
         0   125.0000   115.0000         0         0
   88.0000         0    97.7778   125.0000    9.2222
  220.0000         0         0         0         0
```

Toto řešení je opět odlišné od předchozích.

Jako poslední si ukážeme řešení, kde počítáme rozdíly z cen $c_{i,j}$, a to jak pro sloupce tak i pro řádky. Zadáme tedy

```
[x,na,nb,ha,hb]=VAM(a,b,c,d,1,0);
[x,c,d,iBx]=rovnice(x,c,d,na,nb,ha,hb)
```

a dostáváme řešení

```
x =
         0   125.0000   115.0000         0         0
   88.0000         0    97.7778  125.0000    9.2222
  220.0000         0         0         0         0
```

Toto řešení je stejné jako při výpočtu rozdílů z cen $c_{i,j}$ jen pro sloupce.

3.6 Nalezení optimálního řešení

Optimální řešení můžeme hledat simplexovou metodou odvozenou z KKT podmínek nebo metodou, která je odvozena z teorie duality a využívá speciální struktury úlohy, a která je výhodnější především pro ruční výpočet.

3.6.1 Odvození postupu pro výpočet optimálního řešení z KKT podmínek

Nejprve si zavedeme následující označení

B - bázová matice. Obsahuje pouze vybrané sloupce matice **A** odpovídající bázovým prvkům.

N - nebázová matice. Je to doplněk bázové matice **B**.

$\mathcal{B}$ - horní index označující bázové prvky. Např. vektor $\mathbf{x}^{\mathcal{B}}$ je vektor bázových prvků, vektor $\mathbf{c}^{\mathcal{B}}$ je vektor cen příslušný bázovým prvkům atd.

$\mathcal{N}$ - horní index označující nebázové prvky.

v - vektor duálních proměnných, $\mathbf{v} = (u_1, u_2, ..., u_m, v_1, v_2, ..., v_n)^T$.

$\mathbf{a}_{p,q}$ - vektor odpovídající sloupci matice **A** příslušný prvku $x_{p,q}$. V našem případě se bude jednat o vektor proměnné vstupující do báze.

Při odvozování postupu jsem vycházela z [1]. Vyjdeme z KKT podmínek pro distribuční úlohu (3.2) v rovnicovém tvaru.

$$c_{i,j} - u_i - d_{i,j}v_j - s_{i,j} = 0 \qquad i \in \mathcal{I}; j \in \mathcal{J} \tag{3.7}$$

$$\sum_{j=1}^{n+1} x_{i,j} = a_i \qquad i \in \mathcal{I}; \tag{3.8}$$

$$\sum_{i=1}^{m} d_{i,j}x_{i,j} = b_j \qquad j \in \mathcal{J} \setminus \{n+1\} \tag{3.9}$$

$$x_{i,j} \geq 0 \qquad i \in \mathcal{I}; j \in \mathcal{J} \tag{3.10}$$

$$s_{i,j} \geq 0 \qquad i \in \mathcal{I}; j \in \mathcal{J} \tag{3.11}$$

$$x_{i,j}s_{i,j} = 0 \qquad i \in \mathcal{I}; j \in \mathcal{J} \tag{3.12}$$

kde $\mathcal{I} = \{1, 2, ..., m\}$ a $\mathcal{J} = \{1, 2, ..., n, n+1\}$.

Tyto podmínky můžeme zapsat v maticovém tvaru

$$\mathbf{c^T} - \mathbf{v^T A} - \mathbf{s} = \mathbf{0} \tag{3.13}$$

$$\mathbf{Ax} = \mathbf{b} \tag{3.14}$$

$$\mathbf{x} \geq 0 \tag{3.15}$$

$$\mathbf{s} \geq 0 \tag{3.16}$$

$$x_{i,j}s_{i,j} = 0 \qquad i \in \mathcal{I}; j \in \mathcal{J} \tag{3.17}$$

Předpokládejme, že $\mathbf{x}$ je bázové řešení úlohy (3.2). Ověřme, zda splňuje KKT podmínky. Jelikož je $\mathbf{x}$ bázové řešení, podmínky (3.8) a (3.9) tedy i podmínky (3.14) a (3.15) jsou splněny. Podmínku (3.14) lze také přepsat jako

$$\mathbf{Ax} = \mathbf{Bx}^{\mathcal{B}} + \mathbf{Nx}^{\mathcal{N}} = \mathbf{b}$$

Z definice bázového bodu plyne, že $\mathbf{x}^{\mathcal{N}} = \mathbf{0}$, platí tedy $\mathbf{Bx}^{\mathcal{B}} = \mathbf{b}$.

Dále hledejme vektor $\mathbf{s}$, který bude splňovat podmínku komplementarity (3.17). Pro nebázové prvky $\mathbf{x}^N = 0$ je tato podmínka splněna. Pro bázové prvky $\mathbf{x}^{\mathcal{B}} > 0$ musí tedy platit, že $\mathbf{s}^{\mathcal{B}} = 0$.

Podmínku (3.13) lze rozepsat do dvou rovnic

$$\left(\mathbf{c}^{\mathcal{B}}\right)^T - \mathbf{v^T B} - \mathbf{s}^{\mathcal{B}} = 0$$
$$\left(\mathbf{c}^{\mathcal{N}}\right)^T - \mathbf{v^T N} - \mathbf{s}^{\mathcal{N}} = 0$$

Z předchozího víme, že $\mathbf{s}^B = 0$ a tedy pro bázové prvky platí $\mathbf{c}^{\mathcal{B}} = \mathbf{v^T B}$. Pro nebázové prvky pak vypočítáme

$$s^{\mathcal{N}}_{i,j} = c^{\mathcal{N}}_{i,j} - u_i - d_{i,j}v_j$$

Stále nám chybí splnit podmínku (3.16). Víme, že pro bázové prvky je podmínka (3.16) splněna jelikož $s^{\mathcal{B}}_{i,j} = 0$. Takže nám zbývá splnit $s^{\mathcal{N}}_{i,j} \geq 0$. Pokud toto platí, $\mathbf{x}$ je optimální řešení. Pokud by některý nebázový prvek podmínku porušoval, řešení není optimální a musíme hledat jiné bázové řešení.

Odvodili jsme postup pro ověření optimality splněním KKT podmínek, nyní se podíváme na přechod k nové bázi.

Budeme tedy hledat nový bázový bod. Vybereme tedy index kde $s^{\mathcal{N}}_{p,q} < 0$, který nám vstoupí do báze. Vezmeme bod, kde je podmínka (3.11) nejvíce porušena, bereme tedy minimum ze všech $s^{\mathcal{N}}_{i,j} < 0$. Přechodem k nové bázi se nám změní matice $\mathbf{B}$ a $\mathbf{N}$ a vektory $\mathbf{x}$ a $\mathbf{s}$. Označme si $\hat{\mathbf{x}}$ novou iteraci vyhovující následujícím podmínkám. Změnu provedeme tak, aby platilo, že:

1. hodnota $\hat{x}_{p,q}$ byla kladná a to taková, aby se jedna složka $\mathbf{x}^{\mathcal{B}}$ vynulovala, ale ostatní složky zůstaly kladné (nezáporné u degenerované úlohy)
2. všechny nebázové prvky x^N zůstanou nulové až na $\hat{x}_{p,q}$
3. $\mathbf{x}^B$ se změní na $\hat{\mathbf{x}}^B$ tak, aby stále platilo $\mathbf{A}\hat{\mathbf{x}} = \mathbf{b}$

Ze třetí podmínky nám plyne

$$\mathbf{A}\hat{\mathbf{x}} = \mathbf{B}\hat{\mathbf{x}}^B + \mathbf{a}_{p,q}\hat{x}_{p,q} = \mathbf{b} = \mathbf{A}\mathbf{x} = \mathbf{B}\mathbf{x}^B$$

Odtud tedy dostáváme

$$\hat{\mathbf{x}}^B = \mathbf{x}^B - \mathbf{B}^{-1}\mathbf{a}_{p,q}\hat{x}_{p,q}$$

Pro zjednodušení zavedeme označení $\bar{\mathbf{e}}_{p,q}$ pro $\mathbf{B}^{-1}\mathbf{a}_{p,q}$.

Požadujeme, aby platilo $\hat{\mathbf{x}}^B \geq 0$ a aby existovaly indexy r, s tak, aby platilo $\hat{x}^B_{r,s} = 0$ a $\hat{x}^B_{i,j} \geq 0$ pro $i \neq r$ a $j \neq s$. Musí tedy platit současně

$$\begin{aligned} \hat{x}^B_{r,s} &= x^B_{r,s} - \bar{e}^k_{p,q}\hat{x}_{p,q} = 0 \\ \hat{x}^B_{i,j} &= x^B_{i,j} - \bar{e}^k_{p,q}\hat{x}_{p,q} \geq 0 \qquad\qquad i \neq r,\ j \neq s \end{aligned}$$

kde index k je pořadí příslušného prvku $\hat{x}^B_{r,s}$ nebo $\hat{x}^B_{i,j}$ v bázi. Z tohoto plyne, že hodnota

$$\hat{x}_{p,q} = \min_{\bar{e}^k_{p,q} > 0} \frac{x^B_{i,j}}{\bar{e}^k_{p,q}}$$

3.6.2 Maticový výpočet optimálního řešení

Uvedeme si algoritmus pro výpočet optimálního řešení, který jsme odvodili z KKT podmínek. Tento algoritmus můžeme nalézt v [5].

Algoritmus 3.7.

1. *Otestujeme optimalitu řešení.*

   (a) *Vypočteme duální proměnné* u_i *a* v_j *ze soustavy rovnic*

   $$\mathbf{v^T B} = \mathbf{c}^B$$

   (b) *Dále vypočteme koeficienty* $s_{i,j}$ *pro nebázové prvky užitím vzorců*

   $$s_{i,j} = \begin{cases} c_{i,j} - u_i - d_{i,j}v_j & pro\, i = 1,2,...,m;\ j = 1,2,...,n \\ c_{i,j} - u_i & pro\, j = n+1 \end{cases}$$

   Mezi těmito hodnotami hledáme minimum, které označíme $s_{p,q} = \min\{s_{i,j}\}$.

   (c) *Je-li* $s_{p,q} \geq 0$, *naše řešení je optimální a výpočet končí*

   (d) *Je-li* $s_{p,q} < 0$, *řešení není optimální a proměnná* $x_{p,q}$ *je proměnou vstupující do báze (tj. výběr vstupující proměnné realizujeme Dantzigovým pravidlem minimálního relativního ocenění).*

2. *Určíme vektor proměnné vystupující z báze*

   (a) *Vypočteme vektor* $\bar{\mathbf{e}}_{p,q}$ *z rovnice*

   $$\mathbf{a}_{p,q} = \mathbf{B}\bar{\mathbf{e}}_{p,q}$$

   (b) *Vypočteme* $\delta_{p,q}$ *pro kladné složky vektoru* $\bar{e}_{p,q}$ *podle vzorce:*

   $$\delta_{i,j}^k = \frac{x_{i,j}^k}{\bar{e}_{p,q}^k} \tag{3.18}$$

   kde horní index k značí k-tý prvek vektoru. Z těchto hodnot vyhledáme

   $$\delta_{r,s} = \min_{\forall k} \delta_{i,j}^k$$

   a $x_{r,s}$ *je vystupující proměnná.*

3. Přepočítáme řešení užitím vzorců

$$\bar{x}^k_{i,j} = x^k_{i,j} - \delta_{r,s}\bar{e}^k_{p,q} \qquad \text{pro } i \neq r, \quad j \neq s$$
$$\bar{x}_{r,s} = 0$$
$$\bar{x}_{p,q} = \delta_{r,s}$$

4. Vrátíme se ke kroku 1. a postup opakujeme

Pro výpočet optimálního řešení v programu Matlab jsem napsala funkci `optimální` s hlavičkou

```
function [x]=optimalni(iBx,x,c,d,ha,hb)
```

Ve funkci se volají další tři funkce, které si popíšeme postupně níže. Předpokládáme zde, že máme již vypočtené počáteční řešení, jednou z funkcí `szr`, `mc` nebo `VAM` pro výpočet počátečního řešení. Program bude pracovat, dokud nenalezne optimální řešení naší úlohy, kdy na výstup vypíše celkové náklady a optimální řešení `x`, nebo dokud nenalezne řešení, které sice bude optimální, ale které bude obsahovat pomocné proměnné. Takové řešení není optimální řešení naší úlohy a ta jej ani nemá. Funkci voláme příkazem

```
[x]=optimalni(iBx,x,c,d,ha,hb)
```

Funkce obsažena ve funkci `optimální` je funkce `bazova`, která má hlavičku

```
function [c,d,B,bc]=bazova(c,d,iBx,ha,hb)
```

Tato funkce vytvoří bázovou matici `B` a vektor cen bázových prvků `bc`.

Dále je použita funkce `test_optima`, která má hlavičku

```
function [inB,p1,q1,ms,ni]=test_optima(B,bc,iBx,c,d,ha,hb)
```

Funkce otestuje, zda je řešení optimální. Nejprve vypočítá vektor duálních proměnných `uv` a následně vypočítá koeficienty `s(i,j)` pro nebázové prvky, pomocí kterých zjistí, zda je řešení optimální. Dále nám vypočítá indexy `p1` a `q1` proměnné vstupující do báze a její příslušný bázový vektor `inB` (v maticovém algoritmu označen jako $\mathbf{a}_{p,q}$).

Poslední potřebná funkce, která se objeví ve funkci `optimalni`, je funkce `prepocet` s hlavičkou

```
function [x,cena,iBx]=prepocet(B,inB,x,c,p1,q1,iBx,ha,hb)
```

Tato funkce přepočítá staré řešení. Nejprve vyhledá vektor vystupující proměnné `out` (v maticovém výpočtu označeno jako $\bar{\mathbf{e}}_{p,q}$), poté se vypočítají hodnoty `del` (v maticovém výpočtu označen jako $\delta_{i,j}$) pro kladné prvky vektoru `out`. Nakonec se řešení přepočítá. Dostáváme tedy nové řešení `x`, celkovou cenu nákladů `cena` a upravenou matici indikátorů bázových prvků `iBx`.

Příklad. Ukážeme si výpočet příkladu 3.4. Nejprve vytvoříme vektory a matice pro zadání příkladu. Počáteční řešení vypočteme metodou severozápadního rohu a poté vypočítáme optimální řešení. Zadáme

```
a=[240;320;220];
b=[220;150;180;100];
c=[7,6,5,4;6,8,5,3;4,5,6,5];
d=[0.8,1.2,0.8,1;1,0.5,0.9,0.8;0.6,0.8,0.9,1.2];
[x,na,nb,ha,hb]=szr(a,b,c,d);
[x,c,d,iBx]=rovnice(x,c,d,na,nb,ha,hb));
[x]=optimalni(iBx,x,c,d,ha,hb)
```

a dostáváme na výstupu

```
Celkova_cena =
   3.5361e+003
x je optimální řešení
x =
    0.0000  125.0000   15.0000  100.0000         0
  133.3333         0  186.6667         0         0
  144.4444         0         0         0   75.5556
```

Příklad. Dále si tímto programem vypočítáme příklad 3.3, abychom si ověřili, že příklad nemá optimální řešení, jak jsem uváděli na začátku. Zadejme jej následovně

```
a=[160;320;200];
b=[220;150;180;100];
c=[7,6,5,4;6,8,5,3;4,5,6,5];
d=[0.6,1,0.6,0.8;0.8,0.5,0.7,0.6;0.5,0.7,0.7,1];
[x,na,nb,ha,hb]=szr(a,b,c,d);
[x,c,d,iBx]=rovnice(x,c,d,na,nb,ha,hb));
[x]=optimalni(iBx,x,c,d,ha,hb)
```

A na výstupu dostáváme

```
řešení neexistuje a x není optimální řešení úlohy
x =
         0  150.0000   10.0000         0         0
  275.0000         0   45.0000         0         0
         0         0  100.0000  100.0000         0
         0         0   72.5000         0         0
```

3.6.3 Výpočet optimálního řešení užitím algoritmu využívajícího strukturu úlohy

Uvedeme si algoritmus pro výpočet optimálního řešení, který vychází z teorie duality a speciální struktury úlohy. Napišme si tedy duální úlohu k primární úloze (3.6) distribuční úloh, kde se vyskytují doplňkové i pomocné proměnné. Vycházejme opět z duální úlohy (1.8) uvedené na straně 10, která má po úpravě tvar

$$\begin{aligned} \text{maximalizovat} \quad & \sum_{i=1}^{m} u_i a_i + \sum_{j=1}^{n} v_j b_j & & \\ \text{za podmínek} \quad & c_{i,j} - u_i - d_{i,j} v_j \geq 0 & i = 1,2,...,m;\ j = 1,2,...,n & \\ & 0 - u_i \geq 0 & j = n+1 \qquad i = 1,2,...,m & \\ & M - v_j \geq 0 & i = m+1 \qquad j \in \mathcal{K} & \end{aligned} \tag{3.19}$$

V tomto postupu se vyhneme řešení lineárních rovnic, které je při ručním výpočtu pro větší soustavy náročné, a proto je tento algoritmus vhodnější pro ruční počítání. V algoritmu se vyhneme výpočtu duálních proměnných pomocí $\mathbf{v^T B} = \mathbf{c}^B$ a nahradíme jej výpočtem pomocí jednotlivých vzorců. Výpočet koeficientů $s_{i,j}$ pro nebázové prvky se provádí stejně, jako v předchozím postupu. Také výpočet vektoru proměnné vystupující z báze se nahradí postupem využívajícím speciálního tvaru úlohy. A další postup je stejný jako v předchozím postupu.

V následujícím postupu se předpokládá „uspořádání" řešení před každou iterací, viz [5]. Tímto uspořádáním využijeme strukturu úlohy a můžeme použít následující postupy. Řešení uspořádáme algoritmem

Algoritmus 3.8.

1. *Postupně prohledáváme řádky a hledáme řádek, kde je pouze jeden kladný prvek. Tento prvek zapíšeme a označíme jej $x^{-}_{i,j}$ celý tento řádek vyškrtneme a již jej nebereme v úvahu. Po prohledání všech řádků pokračujeme dalším krokem.*

2. *Hledáme sloupce s jediným kladným prvkem a tento prvek vypíšeme s označením $x^{'}_{i,j}$ a vyškrtneme tento sloupec.*

3. *Postup opakujeme, dokud takto nevyškrtneme všechny prvky nebo nenarazíme na cyklus. Cyklus je uzavřený okruh prvků řešení. To znamená, že již nelze vyškrtnout žádný prvek. V tomto případě vyškrtneme (např. sloupcově) libovolný prvek z cyklu a přejdeme znovu ke kroku 1. a postup opakujeme.*

Až máme vypsané všechny kladné prvky $x_{i,j}$ pro $i = 1,2,\ldots,m$ a $j = 1,2,\ldots,n$, vypíšeme doplňkové prvky v $(m+1)$. sloupci, které lze vyškrtnout pouze řádkově a pomocné prvky v $(n+1)$. řádku, které lze vyškrtnout pouze sloupcově.

Tímto dostaneme uspořádaný řetězec prvků, který nazveme uspořádané řešení. Důvody, proč hledáme právě výše popsané uspořádání řešení, budou jasné z postupu výpočtu duálních proměnných.

Odvození výpočtu duálních proměnných u_i a v_j

Uspořádání bázových prvků je v opačném pořadí, než jak budeme vypočítávat duální proměnné u_i a v_j. Jelikož pro bázové proměnné, které jsou doplňkové nebo pomocné, lze duální proměnné u_i a v_j vypočítat přímo z $0 - u_i = 0$ pro doplňkové proměnné a $M - v_j = 0$ pro pomocné proměnné, vypíšeme tyto prvky jako poslední. S hodnotou M budeme pracovat jako s parametrem, jelikož tato hodnota je prohibitivní. Nejdříve začneme osamocenými bázovými proměnnými v nějakém řádku, kde platí $c_{i,j} - u_i - d_{i,j}v_j = 0$. Jelikož v i-tém řádku není jiný bázový prvek, tak je jasné, že u_i vypočítáme z této rovnice, což můžeme ovšem až tehdy, když známe i v_j, tedy na konci výpočtu. Proto taková $x_{i,j}$ zařadíme do řetězce uspořádání a řádek vyškrtneme. Poté hledáme osamocené prvky ve sloupcích, ze kterých se vypočítá příslušné v_j stejným vzorcem $c_{i,j} - u_i - d_{i,j}v_j = 0$, a tedy musí být počítány až po příslušných u_i. A tento postup opakujeme dokud nemáme všechny bázové prvky uspořádány. Z tohoto plynou pro výpočet u_i a v_j následující vzorce.

$$u_i = \begin{cases} 0 & \text{pro } x^{-}_{i,n+1} \\ c_{i,j} - d_{i,j}v_j & \text{pro } x^{-}_{i,j}, \text{ kde } j \neq n+1 \end{cases}$$

$$v_j = \begin{cases} M & \text{pro } x'_{m+1,j} \\ \frac{1}{d_{i,j}}\left(c_{i,j} - u_i\right) & \textit{pro } x'_{i,j}, \text{ kde } i \neq m+1 \end{cases}$$

Pro prvky z cyklu tento postup nelze použít, jelikož prvky nejsou osamoceny a proto ve vzorcích pro výpočet u_i budou chybět hodnoty v_j a obráceně. Předpokládejme, že máme v řešení uzavřený okruh s prvky $x_{i_1,j_1}, x_{i_1,j_2}, x_{i_3,j_2}, x_{i_3j_3}, \ldots, x_{i_rj_1}$. Zde není osamocený prvek a vynecháme libovolný z nich, například sloupcově x_{i_1,j_1} a tím opět můžeme postupovat ve vynechávání jako v řešení bez uzavřeného okruhu. Dostaneme řetězec uspořádaného řešení pro tento uzavřený okruh. Vypíšeme si rovnice pro výpočet u_i a v_j. Pro prvky vyškrtnuté řádkově vztah pro u_i a pro prvky vyškrtnuté sloupcově

vztah pro v_j

$$\begin{aligned} v_{j_1} &= \frac{1}{d_{i_1,j_1}} \left(c_{i_1,j_1} - u_{i_1}\right) \\ u_{i_1} &= c_{i_1,j_2} - d_{i_1,j_2} v_{j_2} \\ v_{j_2} &= \frac{1}{d_{i_3,j_2}} \left(c_{i_3,j_2} - u_{i_3}\right) \\ u_{i_3} &= c_{i_3 j_3} - d_{i_3 j_3} v_{j_3} \\ &\vdots \\ u_{i_r} &= c_{i_r j_1} - d_{i_r j_1} v_{j_1} \end{aligned} \tag{3.20}$$

Jelikož jsou prvky z uzavřeného okruhu lze tuto soustavu lehce vyřešit, jelikož každé dvě po sobě následující rovnice mají jeden stejný prvek. Postupujeme od poslední rovnice, kde si vyjádříme u_{i_r} pomoci v_{j_1} a dosadíme do rovnice nad ní. Stejně tak u další rovnice až dostaneme konečné vyjádření proměnné v_{j_1}.

Odvození výpočtu vektoru $\bar{\mathbf{e}}_{p,q}$

Pro odvození výpočtu $\bar{\mathbf{e}}_{p,q}$ vycházejme z toho, že nelze porušit řádková a sloupcová omezení. Označíme δ prvek vstupující do báze na místě (p, q), a $r_{i,j}$ označme přírůstek k bázovému prvku $x_{i,j}$.

Pro každý řádek musí platit, že změna součtu v omezení odpovídajícímu tomuto řádku musí být nulová, protože nelze porušit omezení. Máme tak

$$\begin{aligned} \sum_{j \in \mathcal{B}} r_{i,j} &= 0 \quad \text{pro } i \neq p \\ \delta + \sum_{j \in \mathcal{B}} r_{i,j} &= 0 \quad \text{pro } i = p \end{aligned}$$

Pro sloupce musí být celkový přírůstek také nulový. Platí tedy

$$\begin{aligned} \sum_{i \in \mathcal{B}} d_{i,j} r_{i,j} &= 0 \quad \text{pro } j \neq q \\ d_{p,q}\delta + \sum_{i \in \mathcal{B}} d_{i,j} r_{i,j} &= 0 \quad \text{pro } j = q \end{aligned}$$

Vydělíme-li rovnice δ a zavedeme substituci $\overline{e}^k_{p,q} = -r_{i,j}/\delta$, kde dolní index u $\overline{e}$ značí, že proměnná $x_{p,q}$ vstupuje do báze a horní index k značí, že $\overline{e}^k_{p,q}$ je příslušná složka k prvku $x_{i,j}$, který je k-tý v řetězci. Indexy i, j a k mají jednoznačný vztah. Dostáváme

rovnice pro řádky

$$\sum_{j\in\mathcal{B}} \overline{e}^k_{p,q} = 0 \qquad \text{pro } i \neq p \tag{3.21}$$
$$\sum_{j\in\mathcal{B}} \overline{e}^k_{p,q} = 1 \qquad \text{pro } i = p$$

a rovnice pro sloupce

$$\sum_{i\in\mathcal{B}} d_{i,j}\overline{e}^k_{p,q} = 0 \qquad \text{pro } j \neq q \tag{3.22}$$
$$\sum_{i\in\mathcal{B}} d_{i,j}\overline{e}^k_{p,q} = d_{p,q} \qquad \text{pro } j = q$$

Tyto rovnice uspořádáme vzestupně podle k. Na počátku řetězce máme prvek, který je na řádku nebo v sloupci osamocen. Je-li tedy prvek samotný v řádku použijeme (3.21), pokud je prvek osamocen ve sloupci použijeme (3.22). Jestliže vypočtenou hodnotu dosadíme do dalších rovnic, kde se vyskytuje a převedeme ji na pravou stranu, odpovídá to přepočtu a_{pq} na straně 54. Předpokládejme příklad, že prvních několik rovnice pro výpočet vektoru $\overline{\mathbf{e}}$ má následující tvar.

$$\begin{aligned}
\overline{e}^1_{p,q} &= a^{i_1}_{p,q} \\
d_{i_2 j_1}\overline{e}^2_{p,q} + d_{i_1 j_1}\overline{e}^1_{p,q} &= a^{j_1+m}_{p,q} \\
\overline{e}^2_{p,q} + \overline{e}^3_{p,q} &= a^{i_2}_{p,q} \\
&\vdots
\end{aligned}$$

Zde je prvek $x_{i_1 j_1}$ osamocen na řádku a tomu odpovídá první rovnice, ze které dostáváme hodnotu pro $\overline{e}^1_{p,q}$. dosazení hodnoty $\overline{e}^1_{p,q}$ odpovídá přepočtu složek $a_{p,q}$. Z první rovnice plyne $a^{i_1}_{p,q} = a^{i_1}_{p,q} - \overline{e}^1_{p,q}$ a dosazením do druhé rovnice dostáváme $d_{i_2 j_1}\overline{e}^2_{p,q} = a^{j_1+m}_{p,q}$, kde ovšem je již nová hodnota $a^{j_1+m}_{p,q} = a^{j_1+m}_{p,q} - d_{i_1 j_1}\overline{e}^1_{p,q}$. A takto postupujeme dále, až vypočteme všechny hodnoty.

Pokud řešení obsahuje uzavřený okruh, tak tento postup nelze použít a musíme postupovat podobným postupem jako u výpočtu duálních proměnných u_i a v_j pro prvky z uzavřeného okruhu. Vypíšeme si rovnice pro výpočet složek vektoru $\overline{\mathbf{e}}$ pro

prvky z uzavřeného okruhu.

$$\begin{aligned} d_{i_1 j_1}\overline{e}^{k_1}_{p,q} + d_{i_2 j_1}\overline{e}^{k_2}_{p,q} &= a^{j_1+m}_{p,q} \\ \overline{e}^{k_1}_{p,q} + e^{k_3} &= a^{i_1}_{p,q} \\ d_{i_1 j_2}\overline{e}^{k_3} + d_{i_3 j_2}\overline{e}^{k_4} &= a^{j_2+m}_{p,q} \\ \overline{e}^{k_4} + \overline{e}^{k_5} &= a^{i_3}_{p,q} \\ &\vdots \\ \overline{e}^{k_r} + \overline{e}^{k_2} &= a^{i_2}_{p,q} \end{aligned} \tag{3.23}$$

Zde si opět povšimneme, že každé dvě po sobě následující rovnice mají vždy jeden společný prvek. Postupujeme od poslední rovnice výše, až dostáváme vyjádření pro $\overline{e}^{k_1}$.

Algoritmus 3.9. *Máme počáteční řešení určené libovolným způsobem popsaným výše v sekci 3.5*

1. ***Uspořádáme řešení dle algoritmu 3.8***

2. ***Otestujeme optimalitu řešení.***

   (a) ***Vypočteme duální proměnné*** u_i ***a*** v_j

   Prvky vyškrtnuté řádkově nám určují u_i a prvky sloupcové nám určují v_j. Postupujeme od posledního prvku uspořádaného řetězce k prvnímu a postupně tímto způsobem vypočteme všechny u_i a v_j podle vzorců

$$u_i = \begin{cases} 0 & \text{pro } x^{-}_{i,n+1} \\ c_{i,j} - d_{i,j}v_j & \text{pro } x^{-}_{i,j},\ \text{kde } j \neq n+1 \end{cases} \tag{3.24}$$

$$v_j = \begin{cases} M & \text{pro } x^{'}_{m+1,j} \\ \frac{1}{d_{i,j}}\left(c_{i,j} - u_i\right) & \text{pro } x^{'}_{i,j},\ \text{kde } i \neq m+1 \end{cases} \tag{3.25}$$

   kde M je dostatečně velká hodnota. S M budeme pracovat jako s parametrem.

   Obsahuje-li řešení uzavřený okruh, vypíšeme si pro prvky z tohoto okruhu rovnice pro výpočet u_i a v_j podle (3.20)*. Postupujeme postupným dosazováním od poslední rovnice k první. Po výpočtu proměnných u_i a v_j pro prvky z uzavřeného okruhu pokračujeme dále výpočtem ostatních proměnných dle předchozího postupu.*

(b) ***Vypočteme koeficienty*** $s_{i,j}$ ***pro nebázové prvky*** *užitím vzorců*

$$s_{i,j} = \begin{cases} c_{i,j} - u_i - d_{i,j}v_j & pro\, i = 1,2,...,m \quad j = 1,2,...,n \\ c_{i,j} - u_i & pro\, j = n+1 \end{cases}$$

nalezneme minimum označené $s_{p,q} = \min\{s_{i,j}\}$.

i. *Je-li* $s_{p,q} \geq 0$ *naše řešení je optimální a výpočet končí*

ii. *Je-li* $s_{p,q} < 0$ *řešení není optimální a proměnná* $x_{p,q}$ *vstoupí do báze*

3. ***Přechod k nové bázi***

(a) ***Výpočet vektoru proměnné vystupující z báze*** $\bar{\mathbf{e}}_{p,q}$

i. *Vypočteme k-tý prvek vektor* $\bar{\mathbf{e}}_{p,q}$ *podle následujícího vzorce*

$$\bar{e}^k_{p,q} = \begin{cases} a^i_{p,q} & pro\ x^{-s}_{i,j} \\ \frac{a^{j+m}_{p,q}}{d_{i,j}} & pro\ x'^{s}_{i,j} \end{cases} \qquad pro\, s = 1,...,n+m \tag{3.26}$$

ii. *Přepočítáme prvky vektoru* $\mathbf{a}_{p,q}$ *a to*

$$\begin{aligned} a^i_{p,q} &= a^i_{p,q} - \bar{e}^s_{p,q} & i \neq m+1 \\ a^{j+m}_{p,q} &= a^{j+m}_{p,q} - d_{i,j}\bar{e}^s_{p,q} & j \neq n+1 \end{aligned} \tag{3.27}$$

iii. *Vrátíme se ke kroku i. a postup opakujeme, dokud nemáme vypočteny všechny složky vektoru* $\bar{\mathbf{e}}_{p,q}$.

Omezení $i \neq m+1$ *a* $j \neq n+1$ *nám říkají, že pro doplňkové a pomocné proměnné se prvky vektoru* $\mathbf{a}_{p,q}$ *nepřepočítávají.*

Pokud řešení obsahuje cyklus, tak jakmile narazíme na první prvek z cyklu, musíme vypočítat složku vektoru $\bar{\mathbf{e}}_{p,q}$ *odpovídající této proměnné. Nejprve si vypíšeme rovnice pro nalezení prvků vektoru* $\bar{\mathbf{e}}_{p,q}$, *a to jen pro prvky z uzavřeného okruhu podobně jako v* (3.23). *Od poslední rovnice postupným dosazováním postupujeme výše, až dostaneme složku vektoru* $\bar{\mathbf{e}}_{p,q}$ *odpovídající prvnímu prvku z uzavřeného řetězce. Poté také přepočítáme složky vektoru* $\mathbf{a}_{p,q}$, *jak je popsáno výše a postup opakujeme podle předchozího postupu.*

(b) *Pro kladné složky vektoru* $\bar{\mathbf{e}}_{p,q}$ *vypočteme* $\delta_{p,q}$ *podle vzorce:*

$$\delta^k_{i,j} = \frac{x^k_{i,j}}{\bar{e}^k_{p,q}} \tag{3.28}$$

Z těchto hodnot vyhledáme

$$\delta_{r,s} = \min_{\forall k} \delta_{i,j}^k$$

a $x_{r,s}$ *je vystupující proměnná.*

(c) ***Přepočet řešení*** *užitím vzorců*

$$\bar{x}_{i,j}^k = x_{i,j}^k - \delta_{r,s}\bar{e}_{p,q}^k \qquad \text{pro } i \neq r, \quad j \neq s$$
$$\bar{x}_{r,s} = 0$$
$$\bar{x}_{p,q} = \delta_{r,s}$$

4. *Vrátíme se ke kroku 1. a postup opakujeme*

Poznámka. Výpočet kroku 3. tohoto algoritmu 3.9 lze pro přehlednost zapisovat do následující tabulky, která bude mít $m + n$ řádků.

Pořadí-k	Báze	$x_{i,j}$	$\mathbf{a}_{p,q}$	$\bar{\mathbf{e}}_{p,q}$	$\delta_{i,j}$

Pořadí nám určuje pořadový index uspořádaného řešení, tedy do tohoto sloupce zapíšeme postupně pořadová čísla 1 až $m + n$. Do sloupečku báze zapíšeme postupně prvky řetězce uspořádaného řešení a do sloupečku $x_{i,j}$ jejich příslušné hodnoty. Do sloupce $\mathbf{a}_{p,q}$ zapíšeme postupně prvky vektoru vstupující proměnné $\mathbf{a}_{p,q}$, což odpovídá příslušnému sloupci matice $\mathbf{A}$ daného prvku tj. na p-tém místě bude jednička a na $(q + m)$-tém místě bude hodnota $d_{p,q}$. Tento vektor se bude postupně přepočítávat. Ostatní hodnoty se vepisují po jejich výpočtu.

Poznámka. Bázové vektory pro doplňkové proměnné $x_{i,n+1}$ mají pouze jeden prvek nenulový a to 1 na i-tém místě. Bázové vektory pro pomocné proměnné $x_{m+1,j}$ mají pouze hodnotu $d_{i,j}$ na $(m + j)$-tém místě.

Příklad. Celý postup si ukážeme na příkladě 3.4. Vezmeme si počáteční řešení získané Vogelovou aproximační metodou. Toto řešení je počáteční řešení naší úlohy, jelikož se zde nevyskytují pomocné proměnné.

$d_{i,j}$ $x_{i,j}$ $c_{i,j}$	pes	kočka	slon	žirafa	doplňkový	
Linka A	0,8 7	1,2 125 6	0,8 5	1 100 4	1 15 0	240
Linka B	1 120 6	0,5 8	0,9 200 5	0,8 3	1 0	320
Linka C	0,6 $\frac{500}{3}$ 4	0,8 5	0,9 6	1,2 5	1 $\frac{160}{3}$ 0	220
	220	150	180	100		

Začneme uspořádáním řešení podle dříve popsaného postupu. Prohledáváme nejdříve řádky s jedním kladným prvkem. Jelikož takový řádek zde není, pokračujeme prohledáváním sloupců. Jako první vyškrtneme druhý sloupec a zapíšeme prvek $x'_{1,2}$ a poté třetí a čtvrtý sloupec a zapíšeme prvky $x'_{2,3}$ a $x'_{1,4}$. Dále prohledáváme řádky. Jelikož dříve vypsané prvky jsme vyškrtli a doplňkové proměnné se vypisují jako poslední, následuje druhý řádek a prvek $x^-_{2,1}$. Poté vyškrtneme první sloupec s prvkem $x'_{3,1}$ a nakonec první a třetí řádek s doplňkovými prvky $x^-_{1,5}$ a $x^-_{3,5}$. Výsledný řetězec uspořádaného řešení je

$$x'_{1,2},\ x'_{2,3},\ x'_{1,4},\ x^-_{2,1},\ x'_{3,1},\ x^-_{1,5},\ x^-_{3,5}$$

Podle algoritmu 3.9 začneme vypočítávat řádková u_i a sloupcová v_j čísla podle (3.24) a (3.25) od posledního prvku v řetězci. Jelikož $x^-_{3,5}$ je doplňková proměnná, položíme $u_3 = 0$. Stejně tak pro $x^-_{1,5}$ získáváme $u_1 = 0$. Z prvku $x'_{3,1}$ dostáváme hodnotu $v_1 = \frac{10}{6}(4-0) = \frac{20}{3}$, z dalšího prvku $x^-_{2,1}$ dostáváme hodnotu $u_2 = 6 - \frac{20}{3} = -\frac{2}{3}$, z $x'_{1,4}$ dostáváme $v_4 = 1\,(4-0) = 4$, z $x'_{2,3}$ dostáváme $v_3 = \frac{10}{9}\left(5+\frac{2}{3}\right) = \frac{170}{27}$ a z prvku $x'_{1,2}$ dostáváme $v_2 = \frac{10}{12}(6-0) = 5$.

Poté vypočítáme $s_{i,j} = c_{i,j} - u_i - d_{i,j}v_j$ pro nebázové prvky. Všechny hodnoty vidíme v tabulce

$d_{i,j}$ $s_{i,j}$ $c_{i,j}$		pes	kočka	slon	žirafa	doplňkový
	$u_i \diagdown v_j$	$\frac{20}{3}$	5	$\frac{170}{27}$	4	
Linka A	0	0,8 $\frac{5}{3}$ 7	1,2 6	0,8 $-\frac{1}{27}$ 5	1 4	1 0
Linka B	$-\frac{2}{3}$	1 6	0,5 $\frac{37}{6}$ 8	0,9 5	0,8 $\frac{7}{15}$ 3	1 $\frac{2}{3}$ 0
Linka C	0	0,6 4	0,8 1 5	0,9 $\frac{1}{3}$ 6	1,2 1 5	1 0

Vidíme, že minimální hodnota $s_{i,j}$ je pro pole $x_{1,3}$ a to $s_{1,3} = -\frac{1}{27}$. Jelikož je tato hodnota záporná, tak toto řešení není optimální a $x_{1,3}$ vstoupí do báze. Přepočítáme řešení, sestrojíme tedy tabulky pro výpočet vystupující proměnné.

V algoritmu 3.7 se proměnné u_i a v_j počítají z $\mathbf{v^T B} = \mathbf{c}^B$, kde

$$v^T = (u_1, u_2, u_3, v_1, v_2, v_3, v_4)$$

$$\mathbf{B} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0,6 & 0 \\ 1,2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0,9 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{c}^B = (6,\ 4,\ 0,\ 6,\ 5,\ 4,\ 0)$$

Přepsáním dostáváme rovnice

$$\begin{aligned} u_1 + 1,2v_2 &= 6 \\ u_1 + v_4 &= 4 \\ u_1 &= 0 \\ u_2 + v_1 &= 6 \\ u_2 + 0,9v_3 &= 5 \\ u_3 + 0,6v_1 &= 4 \\ u_3 &= 0 \end{aligned}$$

Z těchto rovnic, kde máme přímo vyjádřené u_1 a u_3, postupným dosazováním vypočteme ostatní proměnné, což je v podstatě to samé jako při výpočtu algoritmem 3.9.

Vypočteme vektor $\bar{\mathbf{e}}_{1,3}$ dle (3.26). Nejdříve vypočteme jeho první složku. Jelikož je $x'_{1,2}$ vyškrtnuta sloupcově, počítáme $\bar{e}^1_{1,3} = \frac{a^5_{1,3}}{d_{1,2}} = \frac{0}{12/10} = 0$ a přepočítáme složky vektoru $a_{1,3}$. Nová hodnota $a^1_{1,3} = a^1_{1,3} - \bar{e}^1_{1,3} = 1$ a $a^5_{1,3} = a^5_{1,3} - d_{1,2}\bar{e}^1_{1,3} = 0$. Pokračujeme dále s výpočtem $\bar{e}^2_{1,3}$ a následným přepočtem složek vektoru $a_{1,3}$. Pro doplňkové proměnné vypočteme pouze složky vektoru $\bar{\mathbf{e}}_{1,3}$ a složky vektoru $\mathbf{a}_{1,3}$ zůstávají stejné.

Až máme všechny složky vektoru $\bar{\mathbf{e}}_{1,3}$ vypočtené, dopočteme $\delta_{i,j}$ podle (3.18) pro kladné složky, $\delta_{2,3} = \frac{200}{8/9} = 225$, $\delta_{3,1} = \frac{500/3}{40/27} = 112,5$ a $\delta_{1,5} = \frac{15}{1} = 15$.

s	Báze	$x_{i,j}$	$a_{1,3}$	záměna složek $a_{1,3}$					$\bar{e}_{1,3}$	$\delta_{i,j}$
1	$x'_{1,2}$	125	1	1		1			0	–
2	$x'_{2,3}$	200	0		$-\frac{8}{9}$		0		$\frac{8}{9}$	225
3	$x'_{1,4}$	100	0					$-\frac{40}{27}$	0	–
4	$x^-_{2,1}$	120	0				$\frac{8}{9}$	0	$-\frac{8}{9}$	–
5	$x'_{3,1}$	$\frac{500}{3}$	0	0					$\frac{40}{27}$	112,5
6	$x^-_{1,5}$	15	0,8		0				1	15
7	$x^-_{3,5}$	$\frac{160}{3}$	0			0			$-\frac{40}{27}$	–

V algoritmu 3.7 se vektor $\bar{\mathbf{e}}_{1,3}$ počítal z $\mathbf{a}_{p,q} = \mathbf{B}\bar{\mathbf{e}}_{p,q}$, tj. $\mathbf{a}_{1,3} = \mathbf{B}\bar{\mathbf{e}}_{1,3}$. Pokud složky vektoru očíslujeme horním indexem tak, aby nám souhlasily s prvky, jak jej máme v tabulce, dostáváme rovnice

$$\begin{aligned}
\bar{e}^1_{1,3} + \bar{e}^3_{1,3} + \bar{e}^6_{1,3} &= 1\\
\bar{e}^4_{1,3} + \bar{e}^2_{1,3} &= 0\\
\bar{e}^5_{1,3} + \bar{e}^7_{1,3} &= 0\\
\bar{e}^4_{1,3} + 0,6\bar{e}^5_{1,3} &= 0\\
1,2\bar{e}^1_{1,3} &= 0\\
0,9\bar{e}^2_{1,3} &= 0,8\\
\bar{e}^3_{1,3} &= 0
\end{aligned}$$

Z těchto rovnic, kde máme již přímo vyjádřeno $\bar{e}^1_{1,3}$, $\bar{e}^2_{1,3}$ a $\bar{e}^3_{1,3}$, posupným dosazováním lze vypočítat ostatní hodnoty.

Nejmenší hodnota $\delta_{i,j}$ je $\delta_{1,5} = 15$ a tedy pole $x_{1,5}$ vystoupí z báze. Přepočítáme hodnoty $x_{i,j}$. Tam, kde je prvek vektoru $\bar{\mathbf{e}}_{1,3} = 0$, nemusíme hodnotu $x_{i,j}$ přepočítávat. Ostatní prvky přepočítáme, $x_{2,3} = 200 - \frac{8}{9} \cdot 15 = \frac{560}{3}$, $x_{2,1} = 120 + \frac{8}{9} \cdot 15 = \frac{400}{3}$, $x_{3,1} = \frac{500}{3} - \frac{40}{27} \cdot 15 = \frac{1300}{9}$, $x_{3,5} = \frac{160}{3} + \frac{40}{27} \cdot 15 = \frac{680}{9}$. Prvek $x_{1,5} = 0$ jelikož vystupuje z báze a prvek vstupující do báze je $x_{1,3} = 15$.

Nové řešení zapíšeme do tabulky.

$d_{i,j}$ $x_{i,j}$ $c_{i,j}$	pes	kočka	slon	žirafa	doplňkový	
Linka A	0,8 7	1,2 125 6	0,8 15 5	1 100 4	1 0	240
Linka B	1 $\frac{400}{3}$ 6	0,5 8	0,9 $\frac{560}{3}$ 5	0,8 3	1 0	320
Linka C	0,6 $\frac{1300}{9}$ 4	0,8 5	0,9 6	1,2 5	1 $\frac{680}{9}$ 0	220
	220	150	180	100		

Celý postup opakujeme. Nejprve uspořádáme nové řešení:

$$x'_{1,2},\ x'_{1,4},\ x^-_{1,3},\ x'_{2,3},\ x^-_{2,1},\ x'_{3,1},\ x^-_{3,5}$$

Opět vypočítáme okrajová čísla u_i a v_j od posledního prvku a následně dopočítáme čísla $s_{i,j} = c_{i,j} - u_i - d_{i,j}v_j$ pro nebázové prvky. Všechny hodnoty zapíšeme do tabulky.

$d_{i,j}$ $s_{i,j}$ $c_{i,j}$		pes	kočka	slon	žirafa	doplňkový
	$u_i \diagdown v_j$	$\frac{20}{3}$	$\frac{815}{162}$	$\frac{170}{27}$	$\frac{109}{27}$	
Linka A	$-\frac{1}{27}$	0,8 $\frac{46}{27}$ 7	1,2 6	0,8 5	1 4	1 $\frac{1}{27}$ 0
Linka B	$-\frac{2}{3}$	1 6	0,5 $\frac{1993}{324}$ 8	0,9 5	0,8 $\frac{59}{135}$ 3	1 $\frac{2}{3}$ 0
Linka C	0	0,6 4	0,8 $\frac{79}{81}$ 5	0,9 $\frac{1}{3}$ 6	1,2 $\frac{7}{45}$ 5	1 0

Jak vidíme všechny hodnoty $s_{i,j}$ jsou kladné, to znamená, že jsme nalezli optimální řešení naší úlohy ve tvaru

$d_{i,j}$ $x_{i,j}$ $c_{i,j}$	pes	kočka	slon	žirafa	doplňkový	
Linka A	0,8 7	1,2 125 6	0,8 15 5	1 100 4	1 0	240
Linka B	1 $\frac{400}{3}$ 6	0,5 8	0,9 $\frac{560}{3}$ 5	0,8 3	1 0	320
Linka C	0,6 $\frac{1300}{9}$ 4	0,8 5	0,9 6	1,2 5	1 $\frac{680}{9}$ 0	220
	220	150	180	100		

Celkové náklady činí $\frac{31825}{9} = 3536,11$. Toto řešení znamená, že pes se bude vyrábět $\frac{400}{3}$ hodin na lince B a zároveň $\frac{1300}{9}$ hodin na lince C, kočka se bude vyrábět 125 hodin na lince A , slon se bude vyrábět 15 hodin na lince A a $\frac{560}{3}$ hodin na lince B a žirafa se bude vyrábět 100 hodin na lince A. Hodnota doplňkové proměnné nám značí $\frac{680}{9}$ hodin nevyužité kapacity linky C.

Příklad. Ukážeme si, že příklad 3.3 nemá řešení. Vypočítáme počáteční řešení například Vogelovou aproximační metodou a postupujeme ve výpočtu optimálního řešení stejným způsobem jako v předchozím příkladě, až dostaneme řešení v následujícím tvaru.

$d_{i,j}$ $c_{i,j}$	pes	kočka	slon	žirafa	doplňkový	
Linka A	0,6 7	1 150 6	0,6 10 5	0,8 4	1 0	160
Linka B	0,8 $\frac{1200}{7}$ 6	0,5 8	0,7 $\frac{1040}{7}$ 5	0,6 3	1 0	320
Linka C	0,5 4	0,7 5	0,7 100 6	1 100 5	1 0	200
pomocné	1 $\frac{580}{7}$ M	–	–	–		
	220	150	180	100		

Nejdříve uspořádáme řešení

$$x'_{1,2},\ x'_{3,4},\ x^{-}_{1,3},\ x^{-}_{3,3},\ x'_{2,3},\ x^{-}_{2,1},\ x'_{4,1}$$

A vypočítáme řádková u_i a sloupcová v_j čísla a poté pro nebázové prvky vypočítáme $s_{i,j} = c_{i,j} - u_i - d_{i,j}v_j$. Hodnoty zaznamenáme do následující tabulky.

$s_{i,j}$		pes	kočka	slon	žirafa	doplňkový
	$u_i \diagdown v_j$	M	$\frac{1}{7}+\frac{24}{35}M$	$-\frac{10}{7}+\frac{8}{7}M$	$-2+0,8M$	
Linka A	$\frac{41}{7}-\frac{24}{35}M$	$\frac{8}{7}+\frac{3}{35}M$			$-\frac{9}{35}+\frac{8}{175}M$	$-\frac{41}{7}+\frac{24}{35}M$
Linka B	$6-0,8M$		$\frac{27}{14}+\frac{16}{35}M$		$-1,8+0,32M$	$-6+0,8M$
Linka C	$7-0,8M$	$-3+0,3M$	$-2,1+0,32M$			$-7+0,8M$

Zde si všimneme, že $\min\{s_{i,j}\} = s_{1,4} = -\frac{9}{35}+\frac{8}{175}M$, kde máme kladný násobek dostatečně velkého čísla M. Máme tedy minimum z $s_{i,j}$, které je kladné a řešení je tedy optimální. Toto optimální řešení obsahuje pomocnou proměnnou $x_{4,1} = \frac{580}{7}$, a proto to není optimální řešení původní úlohy 3.3 a tato úloha řešení nemá.

Pro výpočet optimálního řešení tímto algoritmem jsem v programu Matlab napsala funkci `rucnioptimalni` s hlavičkou

```
function [x]=rucnioptimalni(iBx,x,c,d,ha,hb)
```

Na vstupu máme proměnné, které dostaneme po výpočtu počátečního řešení některou z dříve uvedených metod. Na výstupu dostáváme matici optimálního řešení.

Poznámka. Jelikož se tento postup používá převážně pro ruční výpočet pojmenovala jsem ji `rucnioptimalni`, abych ji odlišila od funkce, která pracuje podle algoritmu 3.7

V této funkci se postupně volají další funkce. Funkce `usporadani` s hlavičkou

```
function [rx,index,znak,inxokr]=usporadani(x,iBx,ha,hb)
```

která nám vytvoří řetězec uspořádaného řešení. Vektor `rx` obsahuje hodnoty uspořádaného řešení. Matice `index` typu $(ha+hb)\times 2$ obsahuje indexy uspořádaného řešení a vektor `znak` má prvky 1 pro prvky, které jsou vyškrtnuty řádkově a -1 pro prvky, které jsou vyškrtnuty sloupcově. `Inxokr` je matice indexů prvků z uzavřeného okruhu. Funkce `rucnioptimalni` obsahuje také funkci `testoptimality` s hlavičkou

```
function [inB,p1,q1,ms,ni]=...
        testoptimality(x,index,znak,inxokr,c,d,ha,hb)
```

která nám otestuje, zda je řešení optimální a vypočítá nám indexy proměnné vstupující do báze `p1` a `q1`. Dále se volá funkce `prechod` s hlavičkou

```
function [x,cena,iBx]=...
        prechod(x,c,d,inB,index,rx,znak,ha,hb,p1,q1,iBx,inxokr)
```

která vypočítá vektor vystupující proměnné a poté přepočítá řešení.

4 Další speciální úlohy

Vzhledem k rozsáhlosti dosavadního textu si stručně uvedeme pouze další dvě nejčastější speciální úlohy lineárního programování a to přiřazovací problém a úlohu o maximálním toku. Uvedeme si pouze podobu těchto úloh a nebudeme se jimi zabývat důkladně. V lineárním programování nalezneme spoustu speciálních úloh a úloh příbuzných s dopravní úlohou. Jmenujme například dopravní problém oklikou [6], úloha o nejlevnějším maximálním toku [2], dopravní úloha s omezenými přepravními možnostmi [3], kontejnerový dopravní problém, směšovací problém, úloha o dělení materiálu a další.

4.1 Přiřazovací problém

Přiřazovací problém nejčastěji slouží k přidělení pracovníků na pracovní místa, proto je někdy uváděna jako úloha o rozmístění personálu (viz [3]). Máme k dispozici n pracovníků a n pracovních míst se stejnou mzdou. Každý pracovník může obsadit kteroukoli pozici, ale zkušenosti a schopnosti pracovníků se liší. Přínos i-tého pracovníka na j-té pozici označíme $c_{i,j}$. Chceme, aby celkový přínos pracovníků byl co největší. Úlohu zapíšeme ve tvaru

$$\begin{aligned} \text{maximalizovat} & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{i,j}x_{i,j} & \\ \text{za podmínek} & \sum_{j=1}^{n} x_{i,j} = 1 & i = 1,2,...,n \\ & \sum_{i=1}^{n} x_{i,j} = 1 & j = 1,2,...,n \\ & x_{i,j} \geq 0 & i,j \in \{1,2,...,n\} \end{aligned}$$

Hodnoty $x_{i,j}$ mohou nabývat pouze dvou hodnot 0 a 1. V případě když je j-tá pozice obsazena i-tým pracovníkem $x_{i,j} = 1$, v případě neobsazení pozice $x_{i,j} = 0$. Tato úloha může být i ve tvaru, kde hledáme minimum z účelové funkce a to když $c_{i,j}$ nám budou označovat například plat nebo náklady. Úlohy jsou ekvivalentní s dopravním problémem (viz [6]). Jelikož u přiřazovacího problému dochází ke značné degeneraci, tak není vhodné úlohu řešit stejně jako dopravní úlohu, ale slouží k tomu Maďarská metoda. Ta byla v roce 1955 uvedena Haroldem Kuhnem. Metodu pojmenoval na počest maďarským matematiků Dénese Kőniga and Jenőho Egerváryho, jejichž práce dala teoretický základ této metody. V roce 1957 metodu přezkoumal James Munkres a od té doby je tento algoritmus známý také jako Kuhn-Munkresova metoda (viz [11])

Pokud máme několik stejných nebo podobných pozic lze tyto pozice sloučit do jedné

skupiny. Takto můžeme sloučit i pracovníky, kteří mají stejné nebo podobné schopnosti. Poté tuto úlohu zapíšeme ve tvaru

$$\begin{aligned}
\text{maximalizovat} & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{i,j}x_{i,j} & \\
\text{za podmínek} & \sum_{j=1}^{n} x_{i,j} = a_i & i = 1,2,...,m \\
& \sum_{i=1}^{m} x_{i,j} = b_j & j = 1,2,...,n \\
& x_{i,j} \geq 0 &
\end{aligned}$$

kde a_i je počet uchazečů v i-té skupině se stejnými schopnostmi, b_j je počet stejných volných pozic v j-té skupině (viz [6]). Tato formulace přesně odpovídá formulaci dopravní úlohy.

4.2 Úloha o maximálním toku

Tuto úlohu poprvé zformuloval T. E. Harris v roce 1954 jako zjednodušený tok sovětských železnic. Pro tuto úlohu se kromě lineárního programování využívá také teorie grafů. Úloha o maximálním toku se zabývá především toky v sítích jako jsou vodovodní, železniční, komunikační a jiné. Mějme síť, definovanou n uzly (vrcholy) $\mathbf{V}$ a m hranami (oblouky) $\mathbf{E}$. Máme dva speciální uzly, kde 1. uzel je zdroj (označován také písmenem $\mathbf{s}$ z anglického slova source) a n-tý uzel je spotřebič (označován také písmenem $\mathbf{t}$ z anglického target). Každá hrana má přiřazenou kapacitu $b_{i,j}$. Označme $x_{i,j}$ tok z i do j. Úlohu můžeme zapsat

$$\text{maximalizovat} \sum_{i} x_{i,n} - \sum_{j} x_{n,j} \tag{4.1}$$

$$\text{za podmínek} \sum_{i} x_{i,k} - \sum_{j} x_{k,j} = 0 \qquad k = 2,3,...,n-1 \tag{4.2}$$

$$x_{i,j} \leq b_{i,j} \qquad \forall\,(i,j) \in \mathbf{E} \tag{4.3}$$

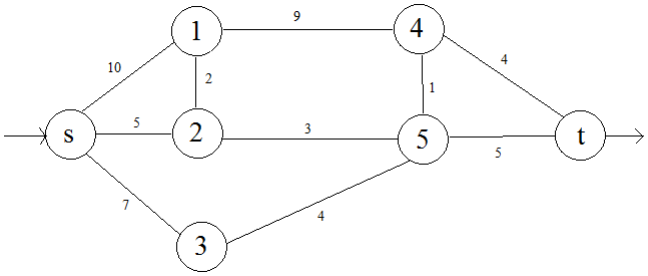
$$x_{i,j} \geq 0 \qquad \forall\,(i,j) \in \mathbf{E} \tag{4.4}$$

kde (4.1) říká, že chceme maximalizovat tok, který do n-tého uzle (spotřebiče) vtéká mínus tok, který z n-tého zdroje vytéká. Podmínka (4.2) říká, že množství, které do každého uzlu (kromě zdroje a spotřebiče) vtéká, musí z uzlu také vytékat. Podmínka (4.3) znamená, že tok mezi dvěma uzly nesmí přesáhnout kapacitu $b_{i,j}$.

V roce 1955 vytvořili Lester R. Ford, Jr. a Delbert R. Fulkerson první algoritmus pro řešení úlohy nazvaný Ford-Fulkersonův algoritmus. Algoritmus je složitosti $O(mf)$,

kde f je maximální tok v grafu. V publikacích je tento algoritmus uváděn nejčastěji. V roce 1972 uvedli Jack Edmonds a Richard Karp algoritmus, který je speciálním případem Ford-Fulkersonova algoritmu, nazvaný Edmons-Karpův algoritmus. Tento algoritmus je složitosti $O(nm^2)$. Nezávisle byl v roce 1970 uveden Dinicův algoritmus formulovaný Yefimem Dinitzem, který má složitost $O(n^2m)$. Viz [12]

Příklad 4.1. Vezmeme si příklad vodovodní sítě naznačený v následujícím schématu. Vrchol označený písmenem s je vodní zdroj a vrchol označený t je konečný vodovodní kohoutek. Ostatní očíslované vrcholy značí spojnice potrubí. U hran, v našem případě je to vodovodní potrubí, jsou zapsány kapacity $b_{i,j}$. Tyto kapacity můžeme chápat jako maximální tok v daném potrubí určeny například průměrem potrubí.



Zapišme si úlohu matematicky

$$\begin{aligned}
&\text{maximalizovat} x_{4,t} + x_{5,t} \\
&\text{za podmínek } (x_{s,1} + x_{2,1} + x_{4,1}) - (x_{1,s} + x_{1,2} + x_{1,4}) = 0 \\
&\qquad (x_{s,2} + x_{1,2} + x_{5,2}) - (x_{2,s} + x_{2,1} + x_{2,5}) = 0 \\
&\qquad (x_{s,3} + x_{5,3}) - (x_{3,s} + x_{3,5}) = 0 \\
&\qquad (x_{1,4} + x_{5,4} + x_{t,4}) - (x_{4,1} + x_{4,5} + x_{4,t}) = 0 \\
&\qquad (x_{2,5} + x_{3,5} + x_{4,5} + x_{t,5}) - (x_{5,2} + x_{5,3} + x_{5,4} + x_{5,t}) = 0 \\
&\qquad x_{s,1} \leq 10 \qquad x_{1,s} \leq 10 \qquad x_{s,2} \leq 5 \qquad x_{2,s} \leq 5 \\
&\qquad x_{s,3} \leq 7 \qquad x_{3,s} \leq 7 \qquad x_{1,2} \leq 2 \qquad x_{2,1} \leq 2 \\
&\qquad x_{1,4} \leq 9 \qquad x_{4,1} \leq 9 \qquad x_{2,3} \leq 3 \qquad x_{3,2} \leq 3 \\
&\qquad x_{3,5} \leq 4 \qquad x_{5,3} \leq 4 \qquad x_{4,5} \leq 1 \qquad x_{5,4} \leq 1 \\
&\qquad x_{4,t} \leq 4 \qquad x_{t,4} \leq 4 \qquad x_{5,t} \leq 5 \qquad x_{t,5} \leq 5 \\
&\qquad x_{i,j} \geq 0 \qquad \forall (i,j) \in \mathbf{E}
\end{aligned}$$

Závěr

Tato práce je zaměřena na řešení dopravní a především distribuční úlohy. Uvedla jsem postupy pro řešení dopravní úlohy a každý postup jsem názorně ukázala na konkrétních příkladech. U distribuční úlohy již nebyl v literaturách uveden jednotný postup, jako tomu bylo u dopravní úlohy. Snažila jsem se tedy uvést postup, který je kompletní a lehce pochopitelný. Za každou uvedenou metodou jsem ji názorně předvedla na řešení konkrétního příkladu. Jak metody pro nalezení počátečního řešení tak metodu pro nalezení optimálního řešení jsem naprogramovala v matematickém softwaru Matlab. Následně jsem těmito programy vypočítala konkrétní příklad, abych ukázala jak příklad zadat a jaký program dává výstup. Všechny metody i programy jsem ukázala na jednom konkrétním příkladě, abychom výsledky mohli srovnat. Díky příkladu nalezeném v [8] jsem zformulovala nutnou podmínku pro existenci řešení. Před samotným řešením jsem se také zabývala řešitelností úlohy. Ukázala jsem převody mezi distribuční úlohou, obecnou distribuční úlohou a dopravní úlohou. Můj cíl nastudovat a naprogramovat postupy pro nalezení distribuční úlohy jsem splnila a navíc jsem se přiučila novým věcem i v jiných oblastech, především v programu Matlab.

Literatura

[1] Jorge Nocedal, Stephen J. Wright, Numerical Optimization,New York, Springer-Verlag, 1999

[2] Plesník J., Dupáčová J., Vlach M., Lineárné programovanie, Bratislava, Alfa-Vydavatelstvo technickej a ekonomickej literatůry, 1990

[3] Gass I. Saul, Lineárne programovanie, Metody a aplikácie, 2.vydání, Bratislava, Slovenské vydavateľstvo technickej literatúry, 1965, přeložil Masaryk Ján, Šimkovic Ján

[4] Do B. Chuong, Convex Optimization Overview, Stanford Univ.: 2009, [cit. 26.11.2012], <http://cs229.stanford.edu/section/cs229-cvxopt2.pdf>

[5] Rychetník L., Zelinka J., Pelzbauerová V., Sbírka příkladů z lineárního programování, 1.vydání, Praha 1, SNTL-Nakladetelství technické literatury,

[6] Dantzig George B., Lineárne programovanie a jeho rozvoj, 1. vydání, Bratislava, Slovenské vydavateľstvo technickej literatúry, 1963(6), přeložil Briška Rudolf

[7] Švrček Jaroslav, Lineární programování v úlohách, Olomouc, Vydavatelství Univerzity Palackého

[8] Kalčevová Jana, Cvičení 4-Formulace úloh LP III. Distribuční úlohy,[online], [cit. 2.12.2011], <http://jana.kalcev.cz/vyuka/kestazeni/4EK311-cv04.pdf>

[9] Goldštejn, E.G, Judin, D.B., Zadači linejnogo programmirovanija transportnovo tipa, Nauka, Moskva 1969

[10] Lagová Milada, Jablonecký Josef, Lineární modely, 1. vydání, Praha : Nakladatelství Oeconomica, 2004,

[11] Hungarian algorithm, [online], [cit. 8.4.2013], <http://en.wikipedia.org/wiki/Hungarian_algorithm>

[12] Maximum flow problem, [online], [cit.8.4.2013], <http://en.wikipedia.org/wiki/Max_flow_problem>

Přílohy

Přílohy se skládají ze dvou částí. Jednou přílohou je CD se všemi programy a druhou výpisy zdrojových kódů funkcí patřících k jednotlivým metodám. Na CD i ve výpisech jsou funkce uvedené na následujícím seznamu.

szr - metoda severozápadního rohu

mc - metoda minimální ceny

VAM - Vogelova aproximační metoda

rovnice - přidání doplňkových a pomocných proměnných

optimalni - výpočet optimálního řešení

bazova - vytvoření bázové matice ve funkci optimalni

test_optima - testování optimality řešení ve funkci optimalni

prepocet - přepočet řešení (přechod k nové bázi) ve funkci optimalni

rucnioptimalni - výpočet optimálního řešení s využitím struktury úlohy

usporadani - uspořádání řešení do uspořádaného řetězce ve funkci rucnioptimalni

testoptimality - testování optimality řešení ve funkci rucnioptimalni

prechod - přepočet řešení (přechod k nové bázi) ve funkci rucnioptimalni

Na CD jsou navíc skripty, ve kterých je zadání řešených úloh.

priklad31 - zadání příkladu 3.2

priklad32 - zadání příkladu 3.3

priklad33 - zadání příkladu 3.4

Metoda severozápadního rohu

```
function [x,na,nb,ha,hb]=szr(a,b,c,d)
%funkce vypočítá počáteční řešení metodou severozápadního rohu
%vstup:a-vektor řádkových omezení
%      b-vektor sloupcových omezení
%      c-matice cenových koeficientů
%      d-matice koeficientů výkonnosti
%výstup:x-matice řešení
%       na-upravený vektor řádkových omezení a
%       nb-uprevený vektor sloupcových omezení b
%       ha-velikost vektoru a
%       hb-velikost vektoru b

na=a;
nb=b;
ha=length(a);
hb=length(b);
x=zeros(ha,hb);
for i=1:1:ha
    for j=1:1:hb
        x(i,j)=min(na(i),nb(j)/d(i,j));  %výpočet x(i,j)
        na(i)=na(i)-x(i,j);              %přepočet a
        nb(j)=nb(j)-(d(i,j)*x(i,j));     %přepočet b
    end
    if na(i)~=0
       i=i+1;
    else
       j=j+1;
    end
end
```

Metoda minimální ceny

```
function [x,na,nb,ha,hb]=mc(a,b,c,d,varianta)
%funkce vypočítá počáteční řešení metodou minimální ceny
%vstup:a-vektor řádkových omezení
%      b-vektor sloupcových omezení
%      c-matice cenových koeficientů
%      d-matice koeficientů výkonnosti
%      varianta-1:hledání minima cen c
%                0:hledání minima z podílů c(i,j)/d(i,j)
%výstup:x-matice řešení
%       na-upravený vektor řádkových omezení a
%       nb-uprevený vektor sloupcových omezení b
%       ha-velikost vektoru a
%       hb-velikost vektoru b

if (nargin<5)
    varianta=0;
end
na=a;
nb=b;
ha=length(a);
hb=length(b);
x=zeros(ha,hb);
I=(1:ha);
J=(1:hb);
if varianta==1
   nc=c;
else
   nc=c./d;
end
while((~isempty(I))&&(~isempty(J)))
      for i=1:ha
          if i~=I
             nc(i,:)=inf;
          end
      end
```

```
    for j=1:hb
        if j~=J
            nc(:,j)=inf;
        end
    end
    hJ=length(J);
    hI=length(I);
    [h,rd]=min((nc));                    %hledání minima
    [h,sl]=min(min((nc)));
    i=rd(sl);
    j=sl;
    x(i,j)=min(na(i),nb(j)/d(i,j));      %výpočet x(i,j)
    na(i)=na(i)-x(i,j);                  %přepočet a
    nb(j)=nb(j)-(d(i,j)*x(i,j));         %přepočet b
    if na(i)==0
       m=find(I==rd(sl));
        I(m)=[];
    else
        n=find(J==sl);
        J(n)=[];
    end
end
```

Vogelova aproximační metoda

```
function [x,na,nb,ha,hb]=VAM(a,b,c,d,varianta,postup)
%funkce vypočítá počáteční řešení jednou z variant
%Vogelovi aproximační metody
%vstup:a-vektor řádkových omezení
%      b-vektor sloupcových omezení
%      c-matice cen
%      d-matice koeficientů výkonnosti
%      varianta -1:hledání minima cen c
%                0:hledání minima z podílů c/d
%      postup -1:výpočet diferencí pro sloupce
%              0:výpočet diferencí pro sloupce i řádky
% výstup:x-matice řešení
%        na-upravený vektor řádkových omezení
%        nb-upravený vektor sloupcových řešení
%        ha, hb-velikost vektoru a,b

if (nargin<5)
   varianta=0;
end
if (nargin<6)
   postup=0;
end
na=a;
nb=b;
ha=length(a);
hb=length(b);
x=zeros(ha,hb);
I=(1:ha);
J=(1:hb);
if varianta==1                    % zvolení nc dle varianty
    nc=c;
else
    nc=c./d;
end
nc1=nc;
```

```
while((~isempty(I))&&(~isempty(J)))
    for i=1:ha
        if i~=I
            nc(i,:)=inf;
        end
    end
    for j=1:hb
        if j~=J
            nc(:,j)=inf;
        end
    end
    [h1,rd1]=min((nc));          %výpočet diferencí pro sloupce
    for j=1:hb
        nc(rd1(j),j)=inf;
    end
    h2=min((nc))
    for j=1:hb
        if j==J(J==j)
            nc(rd1(j),j)=nc1(rd1(j),j);
        end
    end
    dsl=h2-h1;
    if postup==0                  %prohledávání řádků
        [hr1,sl1]=min(nc,[],2);  %výpočet diference pro řádky
        for i=1:ha
            nc(i,sl1(i))=inf;
        end
        hr2=min(nc,[],2);
        for i=1:ha
            nc(i,sl1(i))=nc1(i,sl1(i));
        end
        drd=hr2-hr1;
        dif=[drd',dsl];
        maxdrs=max(dif);
        indexd=find(dif==maxdrs);
        indexdrd=indexd(indexd<=ha);
        indexdsl=indexd(indexd>ha)-ha;
        [minncrd,ird]=min(hr1(indexdrd));
```

```
        [minncsl,isl]=min(h1(indexdsl));
        mnc=min([minncrd,minncsl]);
        if mnc==minncrd          %indexy i a j v VAM i s řádky
            i=indexdrd(ird);
            j=sl1(i);
        elseif mnc==minncsl
            j=indexdsl(isl);
            i=rd1(j);
        end
    else
        mxdl=max(dsl);           %indexy i a j v VAM se sloupci
        sl=find(dsl==mxdl);
        [mcd,isl]=min(h1(sl));
        j=sl(isl);
        i=rd1(j);
    end
    x(i,j)=min(na(i),nb(j)/d(i,j));      %výpočet x(i,j)
    na(i)=na(i)-x(i,j);
    nb(j)=nb(j)-(d(i,j)*x(i,j));
    if na(i)==0                  %vynechání indexů
        m=find(I==i);
        I(m)=[];
    else
        n=find(J==j);
        J(n)=[];
    end
end
```

Přidání doplňkových a pomocných proměnných

```
function [x,c,d,iBx]=rovnice(x,c,d,na,nb,ha,hb)
%funkce přidá doplňkové a pomocné proměnné
%vstup:x-matice řešení
%      c-matice cen
%      d-matice koeficientů výkonnosti
%      na-upravený vektor rádkových omezení a
%      nb-upravený vektor sloupcových omezení b
%      ha-velikost vektoru a
%      hb-velikost vektoru b
%výstup:x-nová matice řešení
%       c-matice cen
%       d-matice koeficientů výkonnosti
%       iBx-matice indikátorů bázových prvků x

x(1:ha,hb+1)=0;
c(1:ha,hb+1)=0;
d(1:ha,hb+1)=1;
for i=1:ha                  %přidání doplňkových proměnných
    if na(i)>0
       x(i,hb+1)=na(i);
    end
end
for j=1:hb                  %přidání pomocných proměnných
    if nb(j)>0
       x(ha+1,j)=nb(j);
       c(ha+1,j)=1i;
       d(ha+1,j)=1;
    end
end
[hxa,hxb]=size(x);
iBx=zeros(hxa,hxb);
iBx(x>0)=1;
```

```
klx=x(x>0);
%pokud počáteční řeš. je degenerované přidá se 1 bázový prvek
if length(klx)<ha+hb
    dx=0;
    i=1;
    while (dx==0)&&(i<=ha)
        sl=find(x(i,:)>0);
        if length(sl)==1
            rd=find(x(1:end,sl)>0);
            if length(rd)==1
                for e=1:ha
                    sl2=find(x(e,:)>0);
                    if (length(sl2)==1)&&(sl2~=sl)
                        iBx(e,sl)=1;
                        dx=1;
                    end
                end
                if dx~=1
                    for e=1:hxb
                        rd2=find(x(1:end,e)>0);
                        if (length(rd2)==1)&&(rd2~=rd)
                            iBx(rd,e)=1;
                        end
                    end
                end
            end
        end
        i=i+1;
    end
end
```

Nalezení optimálního řešení

```
function [x]=optimalni(iBx,x,c,d,ha,hb)
%funkce vypočítá z počátečního řešení optimální řešení,
%pokud existuje
%vstup:iBx-matice indikátorů báových prvků x
%      x-matice počátečního řešení
%      c-matice cen
%      d-matice koeficientů výkonnosti
%      ha-velikost vektoru a
%      hb-velikost vektoru b
%výstup:x-matice optimálního řešení

ms=-1;
ni=1;
[hxa,hxb]=size(x);
while ((real(ms)<0)&&(ni==1))||(imag(ms)<0)
    [c,d,B,bc]=bazova(c,d,iBx,ha,hb);
    [inB,p1,q1,ms,ni]=test_optima(B,bc,iBx,c,d,ha,hb);
    [x,cena,iBx]=prepocet(B,inB,x,c,p1,q1,iBx,ha,hb);
end
if hxa>ha
    if x(ha+1,:)==zeros(1,hxb)
        x(ha+1,:)=[];
    end
end
if ni==1
    Celkova_cena=cena
    fprintf('x je optimální řešení')
else
    fprintf('řešení neexistuje a x není optimální řešení úlohy')
end
```

Vytvoření bázové matice

```
function [c,d,B,bc]=bazova(c,d,iBx,ha,hb)
%funkce vytvoří bázovou matici B a vektor cen bázových prvků
% vstup:c-matice cen
%       d-matice koeficientů
%       iBx-matice indikátorů bázových prvků x
%       ha,hb-velikosti vektorů a,b
% výstup:c-matice cen
%        d-matice koeficientů
%        B-bázová matice
%        bc-vektor cen bázových prvků

[hxa,hxb]=size(iBx);
l=1;
B=zeros(ha+hb,l-1);
for i=1:1:hxa
    for j=1:1:hxb
        if iBx(i,j)>0
            if j==hb+1
                B(i,l)=1;
                bc(l)=c(i,j);
            else
                B(i,l)=1;
                B(ha+j,l)=d(i,j);
                bc(l)=c(i,j);
            end
            if i==ha+1
                B(1:ha+hb,l)=0;
                B(ha+j,l)=d(i,j);
            end
            l=l+1;
        end
    end
end
```

Test optima

```
function [inB,p1,q1,ms,ni]=test_optima(B,bc,iBx,c,d,ha,hb)
%funkce otestuje, zda máme řešení optimální
% vstup: B–bázová matice
%        bc–vektor bázových cen
%        iBx–matice indikátorů bázových prvků x
%        c–matice cen
%        d–matice koeficientů výkonnosti
%        ha–velikost vektoru a
%        hb–velikost vektoru b
%výstup:inB–bázový vektor vstupující proměnné
%       p1–řádkový index vstupující proměnné
%       q1–sloupcový index vstupující proměnné
%       ms–minimální hodnota z s(i,j)
%       ni–indikátor pomocných proměnných

hB=rank(B);
uv=bc/B;                %výpočet vektoru duálních proměnných
u=uv(1:ha);
v=uv(ha+1:ha+hb);
[hxa,hxb]=size(iBx);
for i=1:ha                %výpočet koeficientů s(i,j)
    for j=1:hb+1
        if iBx(i,j)==0
            if j<hxb
                s(i,j)=c(i,j)-u(i)-d(i,j)*v(j);
            else
                s(i,j)=c(i,j)-u(i);
            end
        end
    end
end
```

```
mi=min(min(imag(s)));      %nalezení minima z s(i,j)
ni=all(all(imag(s)==0));
MS1=find(imag(s)==mi);
mr=min(real(s(MS1)));
ms=complex(mr,mi);
[p,q]=find(s==ms);         %určení indexů vstupující proměnné
p1=p(1);
q1=q(1);
inB=zeros(ha+hb,1);
if q1<=hb                  %bázový vektor vstupující proměnné
    inB(p1)=1;
    inB(ha+q1)=d(p1,q1);
else
    inB(p1)=1;
end
```

Přepočet řešení

```
function [x,cena,iBx]=prepocet(B,inB,x,c,p1,q1,iBx,ha,hb)
%funkce přepočítá řešení x
% vstup:B–bázová matice
%       inB–bázový vektor vstupující proměnné
%       x–matice řešení
%       c–matice cen
%       p1–řádkový index vstupující proměnné
%       q1–sloupcový index vstupující proměnné
%       iBx–matice indikátorů bázových prvků v matici x
%       ha–velikost vektoru a
%       hb–velikost vektoru b
% výstup:x–matice nového řešení
%        cena–celková cena nákladů
%        iBx–matice indikátorů bázových prvků v matici x

out=B\inB;                          %výpočet vystupující proměnné
[hxa,hxb]=size(iBx);
l=1;
for i=1:hxa
    for j=1:hxb
        if iBx(i,j)==1
           Bx(l)=x(i,j);            %vektor bázových prvků x
           indexBx(l,1:2)=[i;j];
           l=l+1;
        end
    end
end
hout=length(out);
for l=1:hout
    if out(l)>0
        del(l)=Bx(l)/out(l);        %výpočet hodnot delta
    else
        del(l)=NaN;
    end
end
```

```
t=min(del);
ir=find(del==t,1,'last');
vystupi=indexBx(ir,1);
vystupj=indexBx(ir,2);
for k=1:ha+hb
    i=indexBx(k,1);
    j=indexBx(k,2);
    x(i,j)=x(i,j)-t*out(k);        %přepočet řešení
end
x(p1,q1)=t;
iBx(p1,q1)=1;
iBx(vystupi,vystupj)=0;
for i=1:hxa                        %výpočet celkové ceny
    for j=1:hxb
        CENA(i,j)=x(i,j)*c(i,j);
    end
end
cena=sum(sum(CENA));
```

Výpočet optimálního řešení využitím struktury úlohy

```
function [x]=rucnioptimalni(iBx,x,c,d,ha,hb)
%funkce vypočítá optimální řešení, pokud existuje
%vstup:iBx-matice indikátorů bázových prvků x
%      x-matice řešení
%      c-matice cen
%      d-matice koeficientů výkonnosti
%      ha-velikost vektoru a
%      hb-velikost vektoru b
%výstup:x-matice optimálního řešení

ms=-1;
ni=1;
while ((real(ms)<0)&&(ni==1))||(imag(ms)<0)
    [rx,index,znak,inxokr]=usporadani(x,iBx,ha,hb)
    [inB,p1,q1,ms,ni]=...
        testoptimality(x,index,znak,inxokr,c,d,ha,hb)
    [x,cena,iBx]=...
        prechod(x,c,d,inB,index,rx,znak,ha,hb,p1,q1,iBx,inxokr)
end
[hxa,hxb]=size(x);
if hxa>ha
    if x(ha+1,:)==zeros(1,hxb)
        x(ha+1,:)=[];
    end
end
if ni==1
    Celkova_cena=cena
    fprintf('x je optimální řešení')
else
    fprintf('řešení neexistuje a x není optimální řešení úlohy')
end
```

Uspořádání řešení

```
function [rx,index,znak,inxokr]=usporadani(x,iBx,ha,hb)
%funkce uspořádá řešení
%vstup:x-matice řešení x
%      iBx-matice indikátorů bázových prvků
%      ha-velikost vektoru a
%      hb-velikost vektoru b
%výstup:rx-vektor hodnot uspořádaného řešení
%       index-matice indexů uspořádaného řešení
%       znak-vektor způsobu vynechání prvků řešení
%            1-sloupcově
%            -1-řádkově
%       inxokr-matice indexů prvků z uzavřeného okruhu

[hxa,hxb]=size(x);
nx=x;
niBx=iBx;
rx=[];
inxokr=[];
k=1;
l=1;
il=1;
w=1;
indikokr=0;
while (w>0)
    hrx=length(rx);
    for i=1:ha          %vynechání řádkově
        sl=find(niBx(i,:)>0);
        if length(sl)==1
            if sl~=hb+1
                index(l,1:2)=[i,sl];
                rx(l)=nx(i,sl);
                znak(l)=-1;
                l=l+1;
```

```
            if indikokr==1
                inxokr(il,1:2)=[i,sl];
                il=il+1;
            end
            nx(i,sl)=0;
            niBx(i,sl)=0;
        end
    end
    k=1;
    kl=[];
end
for j=1:hb              %vynechaní sloupcově
   rd=find(niBx(1:end,j)>0);
    if length(rd)==1
        rd=find(niBx(1:end,j)==1);
        if rd~=ha+1
            index(l,1:2)=[rd,j];
            rx(l)=nx(rd,j);
            znak(l)=1;
            l=l+1;
            if indikokr==1
                inxokr(il,1:2)=[rd,j];
                il=il+1;
            end
            nx(rd,j)=0;
            niBx(rd,j)=0;
        end
    end
    k=1;
    kl=[];
end
nnx=niBx(1:ha,1:hb);
khrx=length(rx);
cn=nnx(nnx>0);
w=length(cn);
```

```
    if w>0                  %vynechání prvku z okruhu
        if hrx==khrx
            indikokr=1;
            [rdo,slo]=find(nnx>0,1,'first');
            rx(l)=nx(rdo,slo);
            index(l,1:2)=[rdo,slo];
            znak(l)=1;
            inxokr(il,1:2)=[rdo,slo];
            nx(rdo,slo)=0;
            niBx(rdo,slo)=0;
            il=il+1;
            l=l+1;
        end
    end
end
if hxb>hb                   %vynechání doplňkových proměnných
    for i=1:ha
        if niBx(i,hxb)>0
            rx(l)=nx(i,hxb);
            index(l,1:2)=[i,hxb];
            znak(l)=-1;
            l=l+1;
        end
    end
end
if hxa>ha                   %vynechání pomocných proměnných
    for j=1:hb
        if niBx(hxa,j)>0
            rx(l)=nx(hxa,j);
            index(l,1:2)=[hxa,j];
            znak(l)=1;
            l=l+1;
        end
    end
end
if indikokr==0
    inxokr=1;
end
```

Test optimality

```
function [inB,p1,q1,ms,ni]=...
         testoptimality(x,index,znak,inxokr,c,d,ha,hb)
%funkce otestuje, zda je řešení optimální
%vstup:x-matice řešení
%      index-matice indexů uspořádaného řešení
%      znak-vektor způsobu vynechání prvků řešení
%           1-sloupcověha, hb-hodnost vektoru a, b
%           -1-řádkově
%      inxokr-matice indexů prvků z uzavřeného okruhu
%      c-matice cen
%      d-matice koeficientů výkonnosti
%      ha-velikost vektoru a
%      hb-velikost vektoru b
%Výstup-iBx-matice indikátorů bázových prvků
%       p1-řádkový index vstupující proměnné
%       q1-sloupcový index vstupující proměnné
%       ms-minimum z s
%       ni-indikátor pomocných proměnných

[hxa,hxb]=size(x);
ix=index;
[hokr,h2]=size(inxokr);
if hokr>3                    %výpočet u a v pro okruh
    for k=1:hokr
        i=inxokr(k,1);
        j=inxokr(k,2);
        bcokr(k)=c(i,j);
        Bokr(i,k)=1;
        Bokr(j+ha,k)=d(i,j);
    end
    Bokr=Bokr(any(Bokr,2),:)
    kpul=fix(hokr/2);
    uvokr=bcokr/Bokr;
    ini=inxokr(1:end,1);
    inj=inxokr(1:end,2);
    ui=unique(ini);
```

```
    uj=unique(inj);
    io=[ui,uj];
    for k=1:hokr
        if k<=kpul
            u(io(k))=uvokr(k);
        else
            v(io(k))=uvokr(k);
        end
    end
end
for k=ha+hb:-1:1                    %výpočet u a v
    i=ix(k,1);
    j=ix(k,2);
    if (znak(k)<0)||(j==hb+1)
        if j==hb+1
            u(i)=0;
        else
            u(i)=c(i,j)-d(i,j)*v(j);
        end
    elseif (znak(k)>0)||(i==ha+1)
        if i==ha+1
            v(j)=complex(0,1);
        else
            v(j)=(1/d(i,j))*(c(i,j)-u(i));
        end
    end
end
s=zeros(ha,hxb);                    %výpočet s
p=0;
for i=1:ha
    for j=1:hxb
        for k=1:ha+hb
            if (index(k,1)==i)&&(index(k,2)==j)
                p=1;
                k=ha+hxb;
            end
        end
```

```
        if p~=1
            if j~=hb+1
                s(i,j)=c(i,j)-u(i)-d(i,j)*v(j);
            else
                s(i,j)=c(i,j)-u(i);
            end
        end
        p=0;
    end
end
%nalezeni min z s a indexu vstupujici promenne
mi=min(min(imag(s)));
ni=all(all(imag(s)==0));
MS1=find(imag(s)==mi);
mr=min(real(s(MS1)));
ms=complex(mr,mi);          %minimální hodnota z s(i,j)
[p,q]=find(s==ms);
p1=p(1)                     %indexy vstupující proměnné
q1=q(1)
inB=zeros(ha+hb,1);
if q1<=hb
    inB(p1)=1;
    inB(ha+q1)=d(p1,q1);
else
    inB(p1)=1;
end
```

Přechod k nové bázi

```
function [x,cena,iBx]=...
    prechod(x,c,d,inB,index,rx,znak,ha,hb,p1,q1,iBx,inxokr)
%funkce přepočítá řešení x
%vstup:x-matice řešení
%      c-matice cen
%      d-matice koeficientů výkonnosti
%      inB-vektor vstupující proměnné
%      index-matice indexů bázových prvků
%      rx-vektor hodnot uspořádaného řešení
%      znak-vektor udávající způsob vynechání prvků
%      ha-velikost vektoru a
%      hb-velikost vektoru b
%      p1-řádkový index vstupující proměnné
%      q1-sloupcový index vstupující proměnné
%      iBx-matice indikátorů bázových prvků
%      inxokr-matice indexů prvků z uzavřeného okruhu
%výstup:x-matice nového řešení
%       cena-celková cena nákladů
%       iBx-nová matice indikátorů bázových prvků

[hxa,hxb]=size(x);
in=index;
e=zeros(ha+hb,1);
hinx=length(inxokr);
for k=1:ha+hb          %výpočet vektoru vystupující proměnné e
    i=in(k,1);
    j=in(k,2);
    if (hinx>1)&&(i==inxokr(1,1))&&(j==inxokr(1,2))
        ninB=inB;                          %výpočet pro okruh
        pok=length(inxokr);
        eB=zeros(ha+hb,pok);
```

```
        for k1=1:pok
            i1=inxokr(k1,1);
            j1=inxokr(k1,2);
            eB(i1,k1)=1;
            eB(j1+ha,k1)=d(i1,j1);
        end
        ee=eB\ninB;
        e(k)=ee(1);
    elseif znak(k)<0
        e(k)=inB(i);
    elseif znak(k)>0
        e(k)=inB(j+ha)/d(i,j);
    end
    if (i~=ha+1)&&(j~=hb+1)      %přepočet inB
        inB(i)=inB(i)-e(k)
    end
    if (i~=ha+1)&&(j~=hb+1)
        inB(j+ha)=inB(j+ha)-d(i,j)*e(k)
    end
end
del=zeros(ha+hb,1);
for k=1:ha+hb                    %výpočet delta
    if e(k)>0
        del(k)=rx(k)/e(k);
    else
        del(k)=NaN;
    end
end
hd=min(del);
ir=find(del==hd,1,'last');
vystupi=in(ir,1)          %vystupující proměnná
vystupj=in(ir,2)
for k=1:ha+hb               %přepočet řešení
    i=in(k,1);
    j=in(k,2);
    x(i,j)=x(i,j)-hd*e(k);
end
```

```
x(p1,q1)=hd;
iBx(p1,q1)=1;
iBx(vystupi,vystupj)=0;
for i=1:hxa                    %celkové náklady
    for j=1:hxb
        CENA(i,j)=x(i,j)*c(i,j);
    end
end
cena=sum(sum(CENA));
```