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

DIPLOMOVÁ PRÁCE

Metody vnitřních bodů pro úlohu lineárního programování

Vedoucí diplomové práce:
RNDr. Pavel Ženčák, Ph.D.
Rok odevzdání: 2010

Vypracovala:
Jana Pechová
AME, V. ročník

Prohlášení

Prohlašuji, že jsem diplomovou práci zpracovala samostatně pod vedením RNDr. Pavla Ženčáka, Ph.D. a uvedla jsem všechny použité zdroje informací.

V Olomouci dne 8. dubna 2010 Jana Pechová v.r.

Poděkování

Děkuji všem, kteří přispěli k tvorbě této diplomové práce, jmenovitě RNDr. Pavlu Ženčákovi, Ph.D. za všechny poskytnuté rady a informace.

Obsah

Značení **5**

Úvod **8**

1 Základy LP a metod vnitřních bodů **10**
1.1 Lineárního programování 10
1.1.1 Formulace úlohy lineárního programování a množin jejího řešení . 10
1.1.2 Základní věty teorie duality 11
1.1.3 Podmínky optimality 11
1.2 Primárně-duální metody vnitřních bodů 12
1.3 Centrální cesta 14
1.4 Základní tvar primárně-duálního algoritmu 17

2 Metody sledování cesty **19**
2.1 Metody sledování (centrální) cesty s krátkým krokem 20
2.2 Metody sledování (centrální) cesty s dlouhým krokem 23

3 Algoritmy Mehrotrova typu **29**
3.1 Mehrotrův algoritmus 29
3.1.1 Popis Mehrotrova algoritmu 29
3.1.2 Schéma Mehrotrova algoritmu 32
3.2 Modifikace Mehrotrova algoritmu 33
3.2.1 Modifikace č. 1 34
3.2.2 Modifikace č. 2 35
3.2.3 Modifikace č. 3 37
3.2.4 Modifikace č. 4 39

4 Realizace a numerické testování **42**
4.1 Implementace algoritmů 42
4.1.1 Volba startovacího bodu 42
4.1.2 Stanovení délky kroku 43

4.1.3 Stanovení centrujícího parametru v metodě sledování cesty s dlouhým krokem . . . 44
4.1.4 Stanovení směru . . . 45
4.1.5 Poznámky k modifikacím Mehrotrova algoritmu . . . 46
4.1.6 Ukončovací kriteria . . . 46
4.2 Programové kódy metod . . . 47
4.2.1 Popis funkcí jednotlivých metod . . . 47
4.2.2 Popis volání funkcí . . . 51
4.2.3 Nastavování volitelných parametrů . . . 51
4.3 Numerické výsledky . . . 52
4.3.1 Testovací úlohy . . . 52
4.3.2 Volba parametrů pro testování . . . 52
4.3.3 Výsledky pro základní volbu parametrů . . . 53
4.3.4 Výsledků metod při změnách hodnoty parametru γ . . . 59
4.3.5 Výsledky metod při změně hodnoty parametru τ . . . 72
4.3.6 Výsledky metod při změnách hodnoty parametru δ . . . 84
4.3.7 Výsledků metod při změně hodnoty parametru β . . . 90
4.3.8 Výsledky metod při nejlepších z předložených hodnot parametrů . . . 93
4.3.9 Adaptivní volba parametrů τ a δ . . . 103

Závěr **112**

Značení

$\mathbb{R}^n$	prostor reálných n-rozměrných vektorů.
$\mathbb{R}^{m\times n}$	množina reálných matic typu $m \times n$.
$\mathbf{A}$	matice koeficientů úlohy lineárního programování typu $m \times n$ ($\mathbf{A} \in \mathbb{R}^{m\times n}$) o hodnosti $m \leq n$.
$\mathbf{x}$	n-rozměrný sloupcový vektor primárních proměnných ($\mathbf{x} \in \mathbb{R}^n$).
$\mathbf{y}$	m-rozměrný sloupcový vektor duálních proměnných ($\mathbf{y} \in \mathbb{R}^m$), tj. Lagrangeovy multiplikátory podmínek $\mathbf{Ax} = \mathbf{b}$.
$\mathbf{s}$	n-rozměrný sloupcový vektor přídavných proměnných ($\mathbf{s} \in \mathbb{R}^n$), tj. Lagrangeovy multiplikátory podmínek $\mathbf{x} \geq \mathbf{0}$.
$(\mathbf{x},\mathbf{y},\mathbf{s})$	proměnné v primárně-duální úloze; zápis $(\mathbf{x},\mathbf{y},\mathbf{s})$ představuje zkrácené vyjádření $\left(\mathbf{x}^{\mathrm{T}},\mathbf{y}^{\mathrm{T}},\mathbf{s}^{\mathrm{T}}\right)^{\mathrm{T}}$.
$\mathbf{x}^*$	optimální řešení primární úlohy (1.1) ($\mathbf{x}^* \in \mathbb{R}^n$).
$(\mathbf{y}^*,\mathbf{s}^*)$	optimální řešení duální úlohy (1.2) ($(\mathbf{y}^*,\mathbf{s}^*) \in \mathbb{R}^{m+n}$).
$(\mathbf{x}^*,\mathbf{y}^*,\mathbf{s}^*)$	optimální řešení primárně-duální úlohy.
k	index iterace, $k = 0,1,2,...$
$\left(\mathbf{x}^k,\mathbf{y}^k,\mathbf{s}^k\right)$	k-tá iterace generovaná primárně-duálními metodami.
$(\Delta\mathbf{x},\Delta\mathbf{y},\Delta\mathbf{s})$, resp. $(\Delta\widetilde{\mathbf{x}},\Delta\widetilde{\mathbf{y}},\Delta\widetilde{\mathbf{s}})$	směrový vektor.
$\left(\Delta\mathbf{x}^k,\Delta\mathbf{y}^k,\Delta\mathbf{s}^k\right)$, resp. $\left(\Delta\widetilde{\mathbf{x}}^k,\Delta\widetilde{\mathbf{y}}^k,\Delta\widetilde{\mathbf{s}}^k\right)$	směrový vektor k-té iterace.
$\mathscr{P}$	množina přípustných řešení primární úlohy: $\{\mathbf{x} \mid \mathbf{Ax} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$.
$\mathscr{D}$	množina přípusných řešení duální úlohy: $\left\{(\mathbf{y},\mathbf{s}) \mid \mathbf{A}^{\mathrm{T}}\mathbf{y} + \mathbf{s} = \mathbf{c}, \mathbf{s} \geq \mathbf{0}\right\}$.

$\mathscr{P}^0$ množina striktně přípustných řešení primární úlohy:
$$\{\mathbf{x} \mid \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} > \mathbf{0}\}.$$

$\mathscr{D}^0$ množina striktně přípusných řešení duální úlohy:
$$\left\{(\mathbf{y},\mathbf{s}) \mid \mathbf{A}^{\mathrm{T}}\mathbf{y} + \mathbf{s} = \mathbf{c}, \mathbf{s} > \mathbf{0}\right\}.$$

$\mathscr{F}$ množina přípustných řešení primárně-duální úlohy:
$$\left\{(\mathbf{x},\mathbf{y},\mathbf{s}) \mid \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{A}^{\mathrm{T}}\mathbf{y} + \mathbf{s} = \mathbf{c}, (\mathbf{x},\mathbf{s}) \geq \mathbf{0}\right\}.$$

$\mathscr{F}^0$ množina striktně přípustných řešení primárně-duální úlohy (striktně přípustná množina):
$$\left\{(\mathbf{x},\mathbf{y},\mathbf{s}) \mid \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{A}^{\mathrm{T}}\mathbf{y} + \mathbf{s} = \mathbf{c}, (\mathbf{x},\mathbf{s}) > \mathbf{0}\right\}.$$

α^{pri}, příp. $\widetilde{\alpha}^{pri}$ parametr délky kroku pro primární proměnné, $\alpha^{pri} \in (0,1]$, $\widetilde{\alpha}^{pri} \in (0,1]$.

α^{dual}, příp. $\widetilde{\alpha}^{dual}$ parametr délky kroku pro duální proměnné, $\alpha^{dual} \in (0,1]$, $\widetilde{\alpha}^{dual} \in (0,1]$.

α , příp. α_k parametr délky kroku, $\alpha \in (0,1]$, $\alpha_k \in (0,1]$.

$\mathbf{X}$, příp. $\mathbf{X}^k$ diagonální matice typu $n \times n$ tvořená prvky vektoru $\mathbf{x}$, resp. $\mathbf{x}^k$:
$$\mathbf{X} = diag\,(x_1, x_2, \ldots, x_n)\,, \mathbf{X}^k = diag\left(x_1^k, x_2^k, \ldots, x_n^k\right).$$

$\mathbf{S}$, příp. $\mathbf{S}^k$ diagonální matice typu $n \times n$ tvořená prvky vektoru $\mathbf{s}$, resp. $\mathbf{s}^k$:
$$\mathbf{S} = diag\,(s_1, s_2, \ldots, s_n)\,, \mathbf{S}^k = diag\left(s_1^k, s_2^k, \ldots, s_n^k\right).$$

$\Delta\mathbf{X}$, příp. $\Delta\mathbf{X}^k$ diagonální matice typu $n \times n$ tvořená prvky vektoru $\Delta\mathbf{x}$, resp. $\Delta\mathbf{x}^k$:
$$\Delta\mathbf{X} = diag\,(\Delta x_1, \Delta x_2, \ldots, \Delta x_n),\ \Delta\mathbf{X}^k = diag\left(\Delta x_1^k, \Delta x_2^k, \ldots, \Delta x_n^k\right).$$

$\Delta\widetilde{\mathbf{X}}$, příp.$\Delta\widetilde{\mathbf{X}}^k$ diagonální matice typu $n \times n$ tvořená prvky vektoru $\Delta\widetilde{\mathbf{x}}$, resp. $\Delta\widetilde{\mathbf{x}}^k$:
$$\Delta\widetilde{\mathbf{X}} = diag\,(\Delta\widetilde{x}_1, \Delta\widetilde{x}_2, \ldots, \Delta\widetilde{x}_n),\ \Delta\widetilde{\mathbf{X}}^k = diag\left(\Delta\widetilde{x}_1^k, \Delta\widetilde{x}_2^k, \ldots, \Delta\widetilde{x}_n^k\right).$$

$\Delta\mathbf{S}$, příp. $\Delta\mathbf{S}^k$ diagonální matice typu $n \times n$ tvořená prvky vektoru $\Delta\mathbf{s}$, resp. $\Delta\mathbf{s}^k$:
$$\Delta\mathbf{S} = diag\,(\Delta s_1, \Delta s_2, \ldots, \Delta s_n),\ \Delta\mathbf{S}^k = diag\left(\Delta s_1^k, \Delta s_2^k, \ldots, \Delta s_n^k\right).$$

$\Delta\widetilde{\mathbf{S}}$, příp. $\Delta\widetilde{\mathbf{S}}^k$ diagonální matice typu $n \times n$ tvořená prvky vektoru $\Delta\widetilde{\mathbf{s}}$, resp. $\Delta\widetilde{\mathbf{s}}^k$:
$$\Delta\widetilde{\mathbf{S}} = diag\,(\Delta\widetilde{s}_1, \Delta\widetilde{s}_2, \ldots, \Delta\widetilde{s}_n),\ \Delta\widetilde{\mathbf{S}}^k = diag\left(\Delta\widetilde{s}_1^k, \Delta\widetilde{s}_2^k, \ldots, \Delta\widetilde{s}_n^k\right).$$

$\mathbf{e}$ n-rozměrný jednotkový sloupcový vektor: $\mathbf{e}=(1,1,\ldots,1)^{\mathrm{T}}$.

$log\,(\cdot)$ přirozený logaritmus log_e.

$F\,(\cdot)$ funkce proměnných $(\mathbf{x},\mathbf{y},\mathbf{s})$ tvořená Karush-Kuhn-Tuckerovými podmínkami optimality :
$$F : \mathbb{R}^{2n+m} \to \mathbb{R}^{2n+m}\,.$$

$J\,(\cdot)$ Jakobiho matice funkce F.

$\|\cdot\|_2$, resp. $\|\cdot\|$ Euklidovská norma:
$$\text{pro vektor } \mathbf{u} \in \mathbb{R}^n \text{ definovaná vztahem } \|\mathbf{u}\| = \left(\sum_{i=1}^{n} u_i^2\right)^{\frac{1}{2}}.$$

$\|\cdot\|_\infty$ maximová norma:

pro vektor $\mathbf{u} \in \mathbb{R}^n$ definovaná vztahem $\|\mathbf{u}\|_\infty = \max_{i=1,\ldots,n} |u_i|$.

$\|\cdot\|_1$ jedničková norma:

pro vektor $\mathbf{u} \in \mathbb{R}^n$ definovaná vztahem $\|u\|_1 = \sum_{i=1}^{n} |u_i|$.

C centrální cesta:

$$\{(\mathbf{x}_\iota, \mathbf{y}_\iota, \mathbf{s}_\iota) \mid \iota > 0\}.$$

ι parametr centrální cesty, $\iota > 0$.

σ, příp. σ_k centrující parametr, $\sigma \in [0,1]$, $\sigma_k \in [0,1]$.

μ , příp. μ_k míra duality:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i s_i = \frac{\mathbf{x}^{\mathrm{T}}\mathbf{s}}{n},\ \mu_k = \frac{1}{n}\sum_{i=1}^{n} x_i^k s_i^k = \frac{\left(\mathbf{x}^k\right)^{\mathrm{T}}\mathbf{s}^k}{n}.$$

$\widetilde{\mu}$, příp. $\widetilde{\mu}_k$ míra duality affine scaling směru (hypotetická hodnota μ, resp. μ_k):

$$\widetilde{\mu} = \frac{\left(\mathbf{x}+\widetilde{\alpha}^{pri}\Delta\widetilde{\mathbf{x}}\right)^{\mathrm{T}}\left(\mathbf{s}+\widetilde{\alpha}^{dual}\Delta\widetilde{\mathbf{s}}\right)}{n},\ \widetilde{\mu}_k = \frac{\left(\mathbf{x}^k+\widetilde{\alpha}_k^{pri}\Delta\widetilde{\mathbf{x}}^k\right)^{\mathrm{T}}\left(\mathbf{s}^k+\widetilde{\alpha}_k^{dual}\Delta\widetilde{\mathbf{s}}^k\right)}{n}.$$

r_b , příp. r_b^k primární reziduum:

$$\mathbf{r}_b = \mathbf{A}\mathbf{x} - \mathbf{b},\ \mathbf{r}_b^k = \mathbf{A}\mathbf{x}^k - \mathbf{b}.$$

$\mathbf{r}_c$, příp. $\mathbf{r}_c^k$ duální reziduum:

$$\mathbf{r}_c = \mathbf{A}^{\mathrm{T}}\mathbf{y} + \mathbf{s} - \mathbf{c},\ \mathbf{r}_c^k = \mathbf{A}^{\mathrm{T}}\mathbf{y}^k + \mathbf{s}^k - \mathbf{c}.$$

$\mathcal{N}_2(\varphi)$ okolí centrální cesty:

$$\left\{(\mathbf{x},\mathbf{y},\mathbf{s}) \in \mathcal{F}^0 \mid \|\mathbf{XSe} + \mu\mathbf{e}\|_2 \leq \varphi\mu\right\} \text{ pro } \varphi \in [0,1).$$

$\mathcal{N}_{-\infty}(\gamma)$ okolí centrální cesty:

$$\left\{(\mathbf{x},\mathbf{y},\mathbf{s}) \in \mathcal{F}^0 \mid x_i s_i \geq \gamma\mu \text{ pro } i = 1,2,\ldots,n\right\} \text{ pro } \gamma \in [0,1) .$$

$\mathcal{N}_{-\infty}(\gamma,\delta)$ okolí centrální cesty pro „nepřípustné“ varianty metod:

$$\left\{(\mathbf{x},\mathbf{y},\mathbf{s}) \mid \|(\mathbf{r}_b,\mathbf{r}_c)\| \leq \frac{\|(\mathbf{r}_b^0,\mathbf{r}_c^0)\|}{\mu_0}\delta\mu,\ (\mathbf{x},\mathbf{s}) > \mathbf{0},\ x_i s_i \geq \gamma\mu \text{ pro } i = 1,2,\ldots,n\right\}$$

pro $\gamma \in (0,1)$, $\delta \geq 1$ a startovací bod $\left(\mathbf{x}^0,\mathbf{y}^0,\mathbf{s}^0\right)$.

Úvod

Lineární programování je specifickým případem konvexního matematického programování, které je užíváno v mnoha oblastech lidské činnosti, např. v průmyslové výrobě pro seřízení výrobních linek nebo řezání materiálu, v zemědělství pro vytváření krmných směsí, v logistice pro řízení zásobovacích řetězců, ve finančnictví pro optimalizaci portfolia.

První matematické úvahy týkající se lineárního programování se objevují již koncem 18. století (např. v pracech Fouriera). Ve 30. letech 20. století byly řešeny speciální úlohy lineárního programování - přiřazovací a dopravní problém. Obecná úloha lineárního programování však byla uvedena až v r. 1947 Georgem Dantzingem, který později (r. 1956) představil simplexový algoritmus [3] pro její řešení. Tato dodnes užívaná metoda bezkonkurenčně dominovala v lineárním programování po dobu 40 let, neboť reálné problémy řeší velmi efektivně - počet iterací potřebný k výpočtu je pouze malým násobkem dimenze problému, čemuž metody jiného typu nedokázaly konkurovat.

S rozvojem výpočetní techniky, umožňující řešení úloh velkých rozměrů, se začala zkoumat výpočetní složitost metod po teoretické stránce. Bylo dokázáno, že simplexová metoda může v nejhorších případech vykazovat exponenciální složitost (Klee a Minty [4]). Tento závěr vedl v 70. letech 20. století k zaměření bádání na hledání alternativních metod pro řešení úloh lineárního programování se zaručenou polynomiální výpočetní složitostí.

Jako první byla publikována Kchachianova elipsoidová metoda r. 1979 [5], jež má sice v nejhorším případě polynomiální výpočetní složitost, ale u všech úloh se k ní blíží, a tak nemohla a nemůže v praktických úlohách konkurovat simplexové metodě. Úctyhodným konkurentem Dantzigovy simplexové metody se stala až Karmarkarova projektivní metoda z r. 1984 [6], založená na primármí formulaci, jež měla dobré také teoretické vlastnosti. Tato metoda inspirovala vznik řady metod podobného typu jako např. „affine-scaling“ metody, metody „logaritmických bariér“, metody „redukce potenciálu“ a metody „sledování cesty“, které jsou označovány jako metody vnitřních bodů. Je možné se setkat s různými variantami těchto metod v závislosti na formulaci úlohy, tj. zda vychází primární, duální nebo primárně-duální úlohy. Na začátku 90. let se jako nejefektivnější praktický přístup prosadily verze založené na primárně-duální úloze. Právě z nich je vycházeno také v této práci.

Cílem předkládané práce je představit obecný princip primárně-duálních metod vnitřních bodů pro úlohy lineárního programování, seznámit s metodami sledování cesty (podrobněji

metodou sledování cesty s krátkým krokem a metodou sledování cesty s dlouhým krokem) a Mehrotrovým algoritmem a jeho modifikacemi. Dále zpracovat algoritmy jednotlivých metod do programových kódů v programu MATLAB 7.5.0 (R2007b), otestovat je na sadě testovacích úloh NETLIB LP a porovnat je na základě obdržených výsledků.

Kapitola 1 představuje úvod do problematiky úloh lineárního programování a metod vnitřních bodů. Definuje základní pojmy, z nichž se vychází v následujících kapitolách. Kapitola 2 pojednává o metodách sledování (centrální) cesty. Po seznámení s jejich obecným principem blíže popisuje metodu sledování cesty s krátkým krokem a metodu sledování cesty s dlouhým krokem. Kapitola 3 se zabývá Mehrotrovým algoritmem a jeho modifikacemi. Mehrotrův algoritmus je pro své dobré praktické výsledky užíván velmi často v softwarech určených pro řešení úloh lineárního programování, neexistuje však pro něj konvergenční teorie. To vedlo k vytvoření množství modifikací, v nichž je tento teoretický nedostatek napraven. Poslední kapitola, Kapitola 4, se věnuje implementaci algoritmů z předešlých kapitol a uvádí numerické výsledky získané jejich aplikací na několik testovacích úloh (s řídkou[1] maticí koeficientů). Na jejich základě je provedeno srovnání jednotlivých metod. Programové kódy algoritmů vytvořené v programu MATLAB 7.5.0 (R2007b) jsou uvedeny na přiloženém CD.

[1] Pojem řídká matice není přesně definovaný, ale obvykle se tím chápe, že matice má méně jak 10% nenulových prvků.

Kapitola 1

Základy lineárního programování a metod vnitřních bodů

1.1 Lineárního programování

1.1.1 Formulace úlohy lineárního programování a množin jejího řešení

Úlohy lineárního programování jsou úlohami, jež obsahují pouze lineární vazby (tzn. lineární účelovou (kriteriální) funkci a lineární omezení) a v nichž se hledá vektor reálných proměnných (optimální řešení).

Tato práce vychází ze standardního tvaru úlohy lineárního programování:

Primární úlohou lineárního programování nazýváme úlohu tvaru

$$\begin{array}{rrcl} \textit{minimalizovat} & \mathbf{c}^{\mathrm{T}}\mathbf{x} & & \\ \textit{za podmínek} & \mathbf{A}\mathbf{x} & = \mathbf{b} & , \\ & \mathbf{x} & \geq \mathbf{0} & . \end{array} \tag{1.1}$$

Duální úloha příslušná k uvedené primární úloze má tvar

$$\begin{array}{rrcrcl} \textit{maximalizovat} & \mathbf{b}^{\mathrm{T}}\mathbf{y} & & & & \\ \textit{za podmínek} & \mathbf{A}^{\mathrm{T}}\mathbf{y} & + & \mathbf{s} & = \mathbf{c} & , \\ & & & \mathbf{s} & \geq \mathbf{0} & . \end{array} \tag{1.2}$$

kde

$\mathbf{c}, \mathbf{x}, \mathbf{s} \in \mathbb{R}^n$ jsou n-rozměrné sloupcové vektory,

$\mathbf{b}, \mathbf{y} \in \mathbb{R}^m$ jsou m-rozměrné sloupcové vektory a

$\mathbf{A} \in \mathbb{R}^{m \times n}$ je matice typu $m \times n$ o hodnosti $m \leq n$.

Primární úloha lineárního programování a k ní příslušná duální úloha jsou označovány jako *dvojice duálně sdružených úloh.*

Množina $\mathscr{P} = \{\mathbf{x} \mid \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$ $\left(\mathscr{D} = \left\{(\mathbf{y}, \mathbf{s}) \mid \mathbf{A}^{\mathrm{T}}\mathbf{y} + \mathbf{s} = \mathbf{c}, \mathbf{s} \geq \mathbf{0}\right\}\right)$ se nazývá *množinou přípustných řešení primární (duální) úlohy*, její prvky se nazývají *přípustná řešení příslušné úlohy.*

Množina $\mathscr{P}^0 = \{\mathbf{x} \mid \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} > \mathbf{0}\}$ $\left(\mathscr{D}^0 = \left\{(\mathbf{y}, \mathbf{s}) \mid \mathbf{A}^{\mathrm{T}}\mathbf{y} + \mathbf{s} = \mathbf{c}, \mathbf{s} > \mathbf{0}\right\}\right)$ se nazývá *množinou striktně přípustných řešení primární (duální) úlohy* a její prvky se označují jako *striktně přípustná řešení příslušné úlohy.*

1.1.2 Základní věty teorie duality

Vztahem primární a duální úlohy lineárního programování se zabývá teorie duality, jejíž základní závěry jsou shrnuty do následujících vět:

Věta 1.1. (Slabá věta o dualitě)
Je-li $\mathbf{x}$ primárně přípustné řešení a $(\mathbf{y}, \mathbf{s})$ duálně přípustné řešení úlohy lineárního programování, pak platí $\mathbf{c}^{\mathrm{T}}\mathbf{x} \geq \mathbf{b}^{\mathrm{T}}\mathbf{y}$. Rovnost nastává právě tehdy, když $\mathbf{x}^{\mathrm{T}}\mathbf{s} = 0$.

Důkaz. viz [10], příp. pro maximalizační úlohu [9]. □

Rozdíl $\mathbf{x}^{\mathrm{T}}\mathbf{s} = \mathbf{c}^{\mathrm{T}}\mathbf{x} - \mathbf{b}^{\mathrm{T}}\mathbf{y}$ je označován jako *duální mezera* dvojice přípustných řešení $\mathbf{x}$ a $(\mathbf{y}, \mathbf{s})$.

Pomocí následující věty je možné na základě řešitelnosti jedné z dvojice duálně sdružených úloh získat informaci o řešitelnosti úlohy k ní duální.

Věta 1.2. (Silná věta o dualitě)
Optimální řešení $\mathbf{x}^*$ primární úlohy (1.1) lineárního programování existuje právě tehdy, když existuje optimální řešení $(\mathbf{y}^*, \mathbf{s}^*)$ úlohy k ní duální (1.2), přičemž $\mathbf{c}^{\mathrm{T}}\mathbf{x}^* = \mathbf{b}^{\mathrm{T}}\mathbf{y}^*$.

Důkaz. viz [10], příp. pro maimalizační úlohu [9]. □

1.1.3 Podmínky optimality

Nutnou a postačující podmínku optimality v úlohách lineárního programování zaručují tzv. Karush-Kuhn-Tuckerovy podmínky, jak dokládají následující věty:

Věta 1.3. Vektor $\mathbf{x}^* \in \mathbb{R}^n$ je optimálním řešením primární úlohy (1.1) právě tehdy, když existují vektory $\mathbf{y}^* \in \mathbb{R}^m$ a $\mathbf{s}^* \in \mathbb{R}^n$ takové, že trojice $(\mathbf{x}, \mathbf{y}, \mathbf{s}) = (\mathbf{x}^*, \mathbf{y}^*, \mathbf{s}^*)$ splňuje

$$\begin{aligned} \mathbf{A}^{\mathrm{T}}\mathbf{y} + \mathbf{s} &= \mathbf{c} \quad , \\ \mathbf{A}\mathbf{x} &= \mathbf{b} \quad , \\ \mathbf{x} &\geq \mathbf{0} \quad , \\ \mathbf{s} &\geq \mathbf{0} \quad , \\ x_i s_i &= 0 \quad , \quad i = 1, \ldots, n. \end{aligned} \tag{1.3}$$

Důkaz. viz [2]. □

Vektory $\mathbf{y}^* \in \mathbb{R}^m$ a $\mathbf{s}^* \in \mathbb{R}^n$ jsou nazývané *Lagrangeovými multiplikátory* podmínek $\mathbf{Ax} = \mathbf{b}$ a $\mathbf{x} \geq \mathbf{0}$.

Poslední rovnice ze soustavy (1.3) je nazývána *podmínkou komplementarity*, jež bývá někdy uváděna v alternativním[1] tvaru $\mathbf{x}^{\mathrm{T}}\mathbf{s} = 0$. Celá soustava (1.3) je označována jako *Karush-Kuhn-Tuckerovy podmínky* (zkráceně *KKT podmínky*).

Věta 1.4. Vektor $(\mathbf{y}^*, \mathbf{s}^*) \in \mathbb{R}^m \times \mathbb{R}^n$ je optimálním řešením duální úlohy (1.2) právě tehdy, když existuje vektor $\mathbf{x}^* \in \mathbb{R}^n$ takový, že KKT podmínky (1.3) platí pro $(\mathbf{x}, \mathbf{y}, \mathbf{s}) = (\mathbf{x}^*, \mathbf{y}^*, \mathbf{s}^*)$.

Důkaz. viz [2]. □

Z uvedeného tedy vyplývá, že vektor $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{s}^*)$ řeší systém (1.3) právě tehdy, když $\mathbf{x}^*$ je optimálním řešením primární úlohy (1.1) a $(\mathbf{y}^*, \mathbf{s}^*)$ je řešením duální úlohy (1.2). Vektor $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{s}^*)$ je nazýván *primárně-duálním řešením*.

KKT podmínky je možné za použití funkce $F : \mathbb{R}^{2n+m} \to \mathbb{R}^{2n+m}$ zapsat následovně:

$$\begin{aligned} F(\mathbf{x}, \mathbf{y}, \mathbf{s}) \quad := \quad \begin{pmatrix} \mathbf{A}^{\mathrm{T}}\mathbf{y} + \mathbf{s} - \mathbf{c} \\ \mathbf{Ax} - \mathbf{b} \\ \mathbf{XSe} \end{pmatrix} &= \mathbf{0}, \\ \mathbf{x} &\geq \mathbf{0}, \\ \mathbf{s} &\geq \mathbf{0}, \end{aligned} \tag{1.4}$$

kde

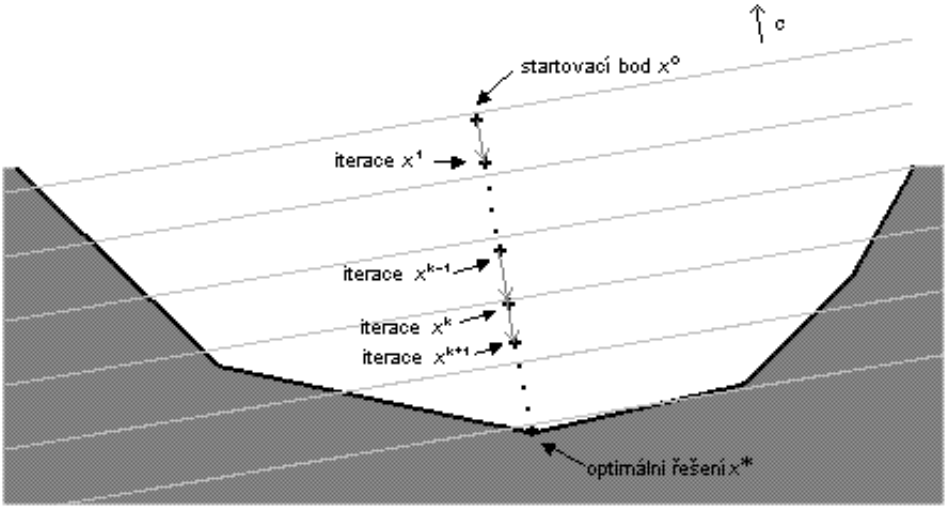
$$\begin{aligned} \mathbf{X} &= diag(x_1, x_2, \ldots, x_n), \\ \mathbf{S} &= diag(s_1, s_2, \ldots, s_n), \\ \mathbf{e} &= (1, 1, \ldots, 1)^{\mathrm{T}} \in \mathbb{R}^n. \end{aligned}$$

1.2 Primárně-duální metody vnitřních bodů

Metody vnitřních bodů patří mezi metody iterační, založené na postupném přibližování se optimálnímu řešení, tzn., že na počátku je zvolen určitý bod (tzv. startovací bod či počáteční aproximace) a odtud se přejde k jinému bodu (tzv. iteraci), který se nachází blíže řešení (viz obrázek 1.1).

Základní primárně-duální metody hledají primárně-duální řešení $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{s}^*)$ pomocí KKT podmínek: aplikují některou z modifikací Newtonovy metody na rovnosti v soustavě (1.3) (příp. rovnost v (1.4)), čímž je nejprve získán směr k další iteraci a poté délka kroku tak, aby byly splněny nerovnosti $\mathbf{x} \geq \mathbf{0}$, $\mathbf{s} \geq \mathbf{0}$. Primárně-duální metody ovšem generují jednotlivé

[1] Uvedený alternativní tvar je možné použít díky požadavkům, aby $\mathbf{x} \geq \mathbf{0}$ a $\mathbf{s} \geq \mathbf{0}$.



Obrázek 1.1: Iterační proces znázorněný v prostoru primárních proměnných $\mathbf{x}$

iterace $(\mathbf{x},\mathbf{y},\mathbf{s})$, které splňují nerovnosti striktně, tedy $\mathbf{x} > \mathbf{0}$, $\mathbf{s} > \mathbf{0}$, leží tedy uvnitř množiny přípustných řešení[2]. Řada z nich také vyžaduje, aby všechny iterace $(\mathbf{x},\mathbf{y},\mathbf{s})$ splňovaly lineární rovnosti primárně-duální úlohy, tj. aby $(\mathbf{x},\mathbf{y},\mathbf{s}) \in \mathscr{F}^0$. Přitom

$$\mathscr{F}^0 = \left\{(\mathbf{x},\mathbf{y},\mathbf{s}) \mid \mathbf{A}\mathbf{x} = \mathbf{b},\, \mathbf{A}^{\mathrm{T}}\mathbf{y} + \mathbf{s} = \mathbf{c},\, (\mathbf{x},\mathbf{s}) > \mathbf{0}\right\}$$

a nazývá se *množinou striktně primárně-duálně přípustných řešení primárně-duální úlohy* (zkráceně *striktně přípustnou množinou*). Definuje se rovněž *množina primárně-duálně přípustných řešení primárně-duální úlohy* (zkráceně *přípustná množina*) jako

$$\mathscr{F} = \left\{(\mathbf{x},\mathbf{y},\mathbf{s}) \mid \mathbf{A}\mathbf{x} = \mathbf{b},\, \mathbf{A}^{\mathrm{T}}\mathbf{y} + \mathbf{s} = \mathbf{c},\, (\mathbf{x},\mathbf{s}) \geq \mathbf{0}\right\}.$$

V Newtonově metodě je řešena rovnost $F(\mathbf{x},\mathbf{y},\mathbf{s}) = \mathbf{0}$ pomocí linearizace kolem aktuálního bodu a směr $(\Delta\mathbf{x},\Delta\mathbf{y},\Delta\mathbf{s})$ je hledán řešením následující soustavy lineárních rovnic:

$$J(\mathbf{x},\mathbf{y},\mathbf{s}) \begin{pmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{y} \\ \Delta\mathbf{s} \end{pmatrix} = -F(\mathbf{x},\mathbf{y},\mathbf{s}), \tag{1.5}$$

kde $J := J(x,y,s)$ je Jakobiho matice funkce F.

[2]Právě tato vlastnost je zachycena v označení „metody vnitřních bodů“.

V případě, že aktuální bod je striktně přípustný, tj.$(\mathbf{x},\mathbf{y},\mathbf{s}) \in \mathscr{F}^0$, jedná se o soustavu

$$\begin{pmatrix} \mathbf{0} & \mathbf{A}^{\mathrm{T}} & \mathbf{I} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{S} & \mathbf{0} & \mathbf{X} \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{y} \\ \Delta\mathbf{s} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ -\mathbf{XSe} \end{pmatrix}. \tag{1.6}$$

Nová iterace v určeném směru je stanovena na základě předpisu

$$(\mathbf{x}+\alpha\Delta\mathbf{x}, \mathbf{y}+\alpha\Delta\mathbf{y}, \mathbf{s}+\alpha\Delta\mathbf{s}), \tag{1.7}$$

kde $\alpha \in (0,1]$ je parametr délky kroku. Délka kroku je určována tak, aby byly splněny podmínky $\mathbf{x}+\alpha\Delta\mathbf{x} > \mathbf{0}$, $\mathbf{s}+\alpha\Delta\mathbf{s} > \mathbf{0}$, proto plný Newtonův krok $(\alpha = 1)$ obvykle není možný.

Metody se základním (nemodifikovaným) Newtonovým směrem (známým také jako *affine scaling směr,* tedy směr orientovaný k optimálnímu řešení, a určeným (1.14)) se obvykle v několika prvních iteracích přibližíí těsně k hranici nezáporného orthantu, v důsledku čehož bývá délka kroku velmi malá $(\alpha \ll 1)$ a konvergence směrem k primárně-duálnímu optimálnímu řešení velmi pomalá. Z tohoto důvodu v praxi využívané primárně-duální metody modifikují základní postup Newtonovy metody následovně:

- upravují směr hledání směrem dovnitř nezáporného orthantu $\mathbf{x} \geq \mathbf{0}$, $\mathbf{s} \geq \mathbf{0}$, následkem čehož je možné učinit delší krok v tomto směru, aniž dojde k porušení podmínek $\mathbf{x} > \mathbf{0}$, $\mathbf{s} > \mathbf{0}$,
- zabraňují složkám $\mathbf{x}$, $\mathbf{s}$, aby se přiblížili „příliš blízko“ k hranicím nezáporného orthantu zavedením vhodných okolí.

1.3 Centrální cesta

Pojem centrální cesta hraje důležitou roli v primárně-duálních algoritmech.

Centrální cesta je definována jako množina bodů $(\mathbf{x}_\iota, \mathbf{y}_\iota, \mathbf{s}_\iota)$, z nichž každý je řešením soustavy

$$\begin{aligned} \mathbf{A}^{\mathrm{T}}\mathbf{y}+\mathbf{s} &= \mathbf{c} \quad , \\ \mathbf{A}\mathbf{x} &= \mathbf{b} \quad , \\ x_i s_i &= \iota \quad , \quad i = 1, ..., n, \\ \mathbf{x} &> \mathbf{0} \quad , \\ \mathbf{s} &> \mathbf{0} \quad , \end{aligned} \tag{1.8}$$

kde $\iota > 0$ je parametr charakterizující příslušnou cestu.

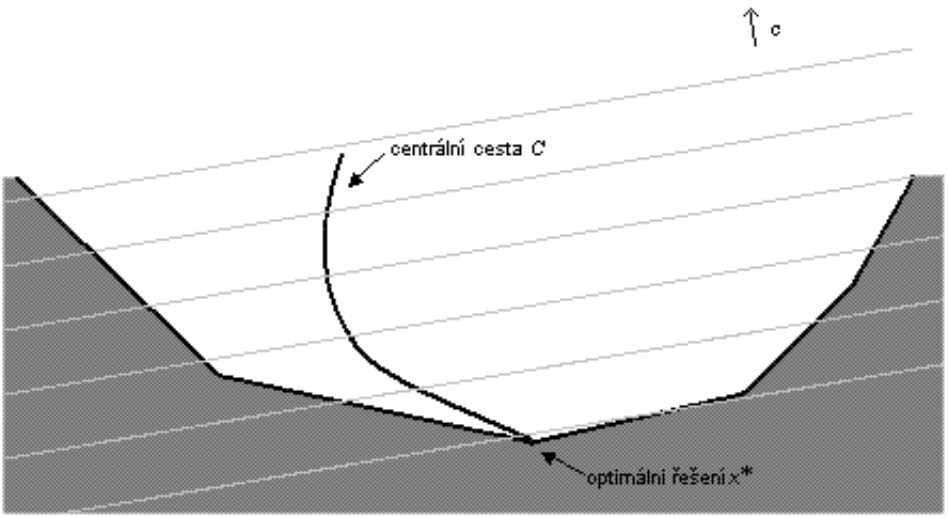
Centrální cesta bývá zapisována

$$C = \{(\mathbf{x}_\iota, \mathbf{y}_\iota, \mathbf{s}_\iota) \mid \iota > 0\}.$$

Pro řešení právě uvedené soustavy (1.8) a tedy i pro body centrální cesty platí:

Pro libovolné $\iota > 0$ existuje bod $(\mathbf{x}_\iota, \mathbf{y}_\iota, \mathbf{s}_\iota)$ právě jeden tehdy a jen tehdy, je-li striktně přípustná množina $\mathscr{F}^0$ neprázdná. Navíc, s klesající hodnotou parametru $\iota > 0$ soustava stále lépe aproximuje soustavu (1.3), tudíž pro $\iota \to 0$ centrální cesta konverguje k primárně-duálnímu řešení úlohy lineárního programování, tj. když $\iota \to 0$, potom $(\mathbf{x}_\iota, \mathbf{y}_\iota, \mathbf{s}_\iota) \to (\mathbf{x}^*, \mathbf{y}^*, \mathbf{s}^*)$.

Centrální cesta je spojnicí bodů, které jsou řešením soustavy (1.8) pro různé hodnoty parametru ι, s optimálním řešením (viz obrázek 1.2). Z obrázku 1.2 je mj. patrná souvislost s bariérovými metodami (viz [11]).



Obrázek 1.2: Centrální cesta znázorněná v prostoru primárních proměnných $\mathbf{x}$

Centrální cesta může být definována, podobně jako KKT podmínky, pomocí funkce $F : \mathbb{R}^{2n+m} \to \mathbb{R}^{2n+m}$ následovně:

$$\begin{aligned} F(\mathbf{x}_\iota, \mathbf{y}_\iota, \mathbf{s}_\iota) &= \begin{pmatrix} \mathbf{A}^{\mathrm{T}}\mathbf{y} + \mathbf{s} - \mathbf{c} \\ \mathbf{A}\mathbf{x} - \mathbf{b} \\ \mathbf{X}\mathbf{S}\mathbf{e} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \iota\mathbf{e} \end{pmatrix}, \\ \mathbf{x}_\iota &> \mathbf{0}, \\ \mathbf{s}_\iota &> \mathbf{0}, \end{aligned} \tag{1.9}$$

kde

$$\begin{aligned} \mathbf{X} &= diag(x_1, x_2, \ldots, x_n), \\ \mathbf{S} &= diag(s_1, s_2, \ldots, s_n), \\ \mathbf{e} &= (1, 1, \ldots, 1)^{\mathrm{T}}. \end{aligned}$$

Soustava rovnic pro určení směru $(\Delta\mathbf{x}, \Delta\mathbf{y}, \Delta\mathbf{s})$ Newtonovou metodou je pak tvaru

$$\begin{pmatrix} \mathbf{0} & \mathbf{A}^{\mathrm{T}} & \mathbf{I} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{S} & \mathbf{0} & \mathbf{X} \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{y} \\ \Delta\mathbf{s} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ -\mathbf{XSe} + \iota\mathbf{e} \end{pmatrix}. \tag{1.10}$$

Pro popis odchylky směru, orientovaného k centrální cestě, získaného řešením právě uvedené soustavy od směru stanoveného řešením (1.6) se zavádí *centrující parametr* $\sigma \in [0,1]$ a *míra duality (parametr duální mezery)* μ definovaná vztahem

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i s_i = \frac{\mathbf{x}^{\mathrm{T}}\mathbf{s}}{n}. \tag{1.11}$$

Parametr centrální cesty se pak určí z $\iota = \sigma\mu$ a soustava (1.10) se změní na tvar

$$\begin{pmatrix} \mathbf{0} & \mathbf{A}^{\mathrm{T}} & \mathbf{I} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{S} & \mathbf{0} & \mathbf{X} \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{y} \\ \Delta\mathbf{s} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ -\mathbf{XSe} + \sigma\mu\mathbf{e} \end{pmatrix}. \tag{1.12}$$

Odtud získaný směr $(\Delta\mathbf{x}, \Delta\mathbf{y}, \Delta\mathbf{s})$ je Newtonův směr orientovaný k bodu centrální cesty $\left(\mathbf{x}_{\sigma\mu}, \mathbf{y}_{\sigma\mu}, \mathbf{s}_{\sigma\mu}\right)$.

Volba parametru $\sigma \in [0,1]$ ovlivňuje směr Newtonova kroku. Pro $\sigma = 1$ je soustavou (1.12) určen centrující směr vedoucí k bodu $\left(\mathbf{x}_\mu, \mathbf{y}_\mu, \mathbf{s}_\mu\right) \in C$. Pohyb tímto směrem obvykle nepřináší žádné nebo velmi malé zmenšení míry duality μ, přivádí však do bodu poměrně daleko od hranice nezáporného orthantu a poskytuje tak možnost pro značné snížení μ v případné další iteraci (krok bude relativně velký). Je-li $\sigma = 0$, ze soustavy (1.12) se stává soustava (1.6), která dává základní Newtonův směr orientovaný k optimálnímu primárně-duálnímu řešení. Algoritmy většinou užívají hodnoty σ z $(0,1)$ jako kompromis mezi zlepšováním centrality a redukcí míry duality .

1.4 Základní tvar primárně-duálního algoritmu

Iterační proces primárně-duálních metod vnitřních bodů vedoucí k optimálnímu primárně-duálnímu řešení je možné popsat pomocí následujícího algoritmu, jež shrnuje poznatky uvedené výše:

Algoritmus PD

Vstup:

startovací bod $\left(\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0\right) \in \mathscr{F}^0$.

begin

for $k = 0, 1, 2, \ldots$

Pro $\sigma_k \in [0,1]$ a $\mu_k = \frac{\left(\mathbf{x}^k\right)^{\mathrm{T}} \mathbf{s}^k}{n}$ najdi řešení soustavy

$$\begin{pmatrix} \mathbf{0} & \mathbf{A}^{\mathrm{T}} & \mathbf{I} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{S}^k & \mathbf{0} & \mathbf{X}^k \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x}^k \\ \Delta\mathbf{y}^k \\ \Delta\mathbf{s}^k \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ -\mathbf{X}^k\mathbf{S}^k\mathbf{e} + \sigma_k\mu_k\mathbf{e} \end{pmatrix}. \tag{1.13}$$

Zvol délku kroku α_k tak, aby $\mathbf{x}^k + \alpha_k\Delta\mathbf{x}^k > \mathbf{0}$ a $\mathbf{s}^k + \alpha_k\Delta\mathbf{s}^k > \mathbf{0}$.

Polož $\left(\mathbf{x}^{k+1}, \mathbf{y}^{k+1}, \mathbf{s}^{k+1}\right) = \left(\mathbf{x}^k + \alpha_k\Delta\mathbf{x}^k, \mathbf{y}^k + \alpha_k\Delta\mathbf{y}^k, \mathbf{s}^k + \alpha_k\Delta\mathbf{s}^k\right)$.

end (for)

end

Uvedený Algoritmus PD, stejně jako úvahy výše, předpokládá, že startovací bod $\left(\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0\right)$ je striktně přípustný, tj. kromě nerovností $\mathbf{x}^0 > \mathbf{0}$ a $\mathbf{s}^0 > \mathbf{0}$ splňuje rovněž lineární rovnice $\mathbf{A}\mathbf{x}^0 = \mathbf{b}$ a $\mathbf{A}^{\mathrm{T}}\mathbf{y}^0 + \mathbf{s}^0 = \mathbf{c}$. V řadě úloh lineárního programování je však obtížné startovací striktně přípustný bod nalézt a někdy je to dokonce nemožné, neboť v některých případech takovýto bod ani neexistuje[3]. Tato skutečnost nepředstavuje žádný problém pro skupinu metod označovanou jako „nepřípustné“ metody vnitřních bodů. Jejich jediným požadavkem na startovací bod je splnění podmínek $\mathbf{x}^0 > \mathbf{0}$ a $\mathbf{s}^0 > \mathbf{0}$. Nepřípustnost počátečního bodu si ovšem vyžádá změnu soustavy pro stanovení směru (1.12) tak, aby byla v každé iteraci zlepšována přípustnost:

$$\begin{pmatrix} \mathbf{0} & \mathbf{A}^{\mathrm{T}} & \mathbf{I} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{S} & \mathbf{0} & \mathbf{X} \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{y} \\ \Delta\mathbf{s} \end{pmatrix} = \begin{pmatrix} -\mathbf{r}_c \\ -\mathbf{r}_b \\ -\mathbf{X}\mathbf{S}\mathbf{e} + \sigma\mu\mathbf{e} \end{pmatrix}, \tag{1.14}$$

kde $\mathbf{r}_b$ a $\mathbf{r}_c$ jsou rezidua definovaná vztahy

[3] zejména v úlohách lineárního programování získaných transformací obecných problémů na standardní tvar

$$\begin{aligned} \mathbf{r}_b &= \mathbf{A}\mathbf{x} - \mathbf{b}, \\ \mathbf{r}_c &= \mathbf{A}^{\mathrm{T}}\mathbf{y} + \mathbf{s} - \mathbf{c}. \end{aligned} \tag{1.15}$$

Směr $(\Delta\mathbf{x}, \Delta\mathbf{y}, \Delta\mathbf{s})$ určený touto soustavou je stále Newtonův směr k bodu $\left(\mathbf{x}_{\sigma\mu}, \mathbf{y}_{\sigma\mu}, \mathbf{s}_{\sigma\mu}\right) \in C.$, vzhledem k reziduím v soustavě (1.15) je ale nyní orientován tak, aby eliminoval nepřípustnost pokud možno hned v prvním kroku. V okamžiku, kdy je délka kroku α rovna 1 (plný krok), rezidua $\mathbf{r}_b$ a $\mathbf{r}_c$ se stanou nulová a všechny následující iterace již jsou striktně přípustné.

Modifikace Algoritmu PD dovolující použití nepřípustného startovacího bodu vypadá následovně:

Algoritmus NPD

Vstup:

startovací bod $\left(\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0\right) : \mathbf{x}^0 > \mathbf{0}, \mathbf{s}^0 > \mathbf{0}$.

begin

for $k = 0, 1, 2, \ldots$

Pro $\sigma_k \in [0,1]$, $\mathbf{r}_c^k = \mathbf{A}^{\mathrm{T}}\mathbf{y}^k + \mathbf{s}^k - \mathbf{c}$, $\mathbf{r}_b^k = \mathbf{A}\mathbf{x}^k - \mathbf{b}$ a $\mu_k = \frac{\left(\mathbf{x}^k\right)^{\mathrm{T}}\mathbf{s}^k}{n}$ najdi řešení soustavy

$$\begin{pmatrix} \mathbf{0} & \mathbf{A}^{\mathrm{T}} & \mathbf{I} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{S}^k & \mathbf{0} & \mathbf{X}^k \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x}^k \\ \Delta\mathbf{y}^k \\ \Delta\mathbf{s}^k \end{pmatrix} = \begin{pmatrix} -\mathbf{r}_c^k \\ -\mathbf{r}_b^k \\ -\mathbf{X}^k\mathbf{S}^k\mathbf{e} + \sigma_k\mu_k\mathbf{e} \end{pmatrix}. \tag{1.16}$$

Zvol délku kroku α_k tak, aby $\mathbf{x}^k + \alpha_k\Delta\mathbf{x}^k > \mathbf{0}$ a $\mathbf{s}^k + \alpha_k\Delta\mathbf{s}^k > \mathbf{0}$.

Polož $\left(\mathbf{x}^{k+1}, \mathbf{y}^{k+1}, \mathbf{s}^{k+1}\right) = \left(\mathbf{x}^k + \alpha_k\Delta\mathbf{x}^k, \mathbf{y}^k + \alpha_k\Delta\mathbf{y}^k, \mathbf{s}^k + \alpha_k\Delta\mathbf{s}^k\right)$.

end (for)

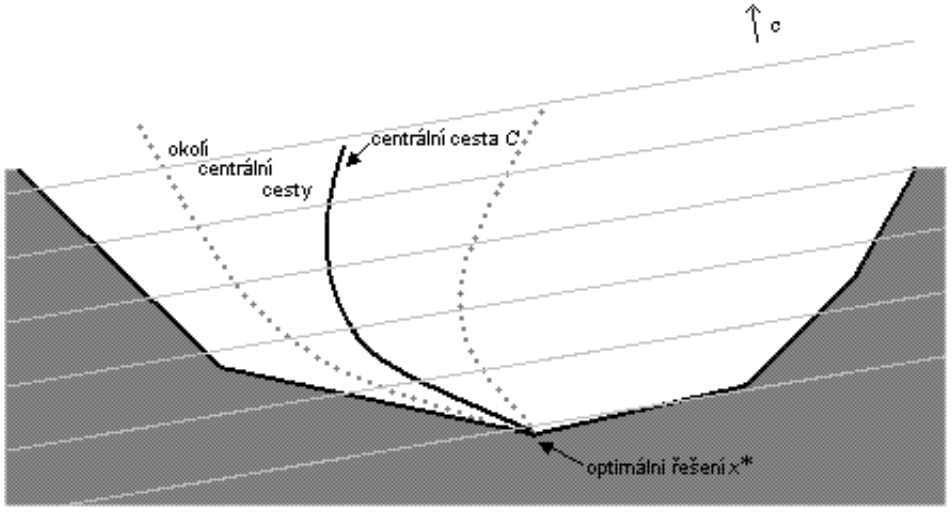
end

Algoritmy jsou ukončovány po splnění jedné nebo několika podmínek. Nejednodušší ukončovací podmínkou je $\mathbf{x}^{\mathrm{T}}\mathbf{s} \leq \varepsilon$, kde ε představuje požadovanou přesnost výpočtu. Cyklus generující iterace se tedy opakuje, dokud není splněna požadovaná podmínka přesnosti. Při odhadu složitosti algoritmů se potom hovoří o ε-přesném řešení. V případě „nepřípustných“ metod bývají rovněž testovány velikosti reziduí. Možný tvar takovýchto ukončovacích podmínek, jakož i alternativu pro podmínku $\mathbf{x}^{\mathrm{T}}\mathbf{s} \leq \varepsilon$, lze nalézt např. v [2].

Kapitola 2

Metody sledování (centrální) cesty

Metody sledování cesty zabezpečují, že se jednotlivé iterace při konvergenci k optimálnímu primárně-duálnímu řešení drží centrální cesty C, nevyžadují však, aby se nacházely přesně na centrální cestě nebo těsně u ní. Stačí, když během poklesu míra duality μ zůstává v určitém jejím okolí (viz obrázek 2.1), a tím se nepřibližují k hranici orthantu $\mathbf{x} > \mathbf{0}$ a $\mathbf{s} > \mathbf{0}$. Za tímto účelem explicitně omezují délku kroku.



Obrázek 2.1: Okolí centrální cesty znázorněné v prostoru primárních proměnných $\mathbf{x}$

Mezi nejvýznamější okolí centrální cesty patří

$$\mathscr{N}_2(\varphi) = \left\{(\mathbf{x},\mathbf{y},\mathbf{s}) \in \mathscr{F}^0 \mid \|\mathbf{XSe} + \mu\mathbf{e}\|_2 \leq \varphi\mu\right\}, \tag{2.1}$$

kde $\varphi \in [0,1)$, které vede na metody sledování cesty s krátkým krokem.

Dále také okolí

$$\mathscr{N}_{-\infty}(\gamma) = \left\{(\mathbf{x},\mathbf{y},\mathbf{s}) \in \mathscr{F}^0 \mid x_i s_i \geq \gamma\mu \text{ pro } i = 1,2,\ldots,n\right\}, \tag{2.2}$$

kde $\gamma \in (0,1)$, vede na metody sledování cesty s dlouhým krokem.
Obvyklé volby parametrů okolí jsou $\varphi = 0.5$ a $\gamma = 10^{-3}$.

Okolí $\mathscr{N}_{-\infty}(\gamma)$ pokrývá téměř celou striktně přípustnou množinu $\mathscr{F}^0$, je-li hodnota γ blízká 0. Naproti tomu okolí $\mathscr{N}_2(\varphi)$ je mnohem více omezené - zahrnuje jen zlomek bodů z $\mathscr{F}^0$ i při volbě parametru φ při jeho horní hranici, tj. blízko 1.

Jednotlivé metody sledování cesty se liší volbou okolí, centrujícího parametru σ a délky kroku α tak, aby zajistily, že každá iterace $(\mathbf{x},\mathbf{y},\mathbf{s})$ bude ležet ve zvoleném okolí. Přitom metody, v nichž se používá okolí $\mathscr{N}_2(\varphi)$, mají teoretickou iterační složitost $O\left(\sqrt{n}\,log\frac{1}{\varepsilon}\right)$. Mezi tyto metody patří metody sledování cesty s krátkým krokem a metody prediktor-korektor[1], které z nich vycházejí. Metody, v nichž se užívá okolí $\mathscr{N}_{-\infty}(\gamma)$, vykazují teoretickou iterační složitost $O\left(n\,log\frac{1}{\varepsilon}\right)$, což je v porovnání s metodami používajícími $\mathscr{N}_2(\varphi)$ horší výsledek.

Implementací voleb okolí a parametrů, příp. dalších úprav do základního tvaru primárně-duálního algoritmu, ať již do „přípustné“ verze (Algoritmus PD) nebo do „nepřípustné“ verze (Algoritmus NPD), jsou obdrženy algoritmy jednotlivých metod sledování cesty.

2.1 Metody sledování (centrální) cesty s krátkým krokem

Metody sledování cesty s krátkým krokem patří mezi nejjednodušší metody vnitřních bodů.

Algoritmus této metody vychází z bodu $\left(\mathbf{x}^0,\mathbf{y}^0,\mathbf{s}^0\right) \in \mathscr{N}_2(\varphi)$. Hodnota centrujícího parametru se volí stejná pro všechny iterace, tedy $\sigma_k \equiv \sigma \in [0,1]$, přičemž se dodržuje platnost vztahu $\sigma = 1 - \frac{\varphi}{\sqrt{n}}$. Největší délka kroku taková, že se následující iterace nachází v okolí $\mathscr{N}_2(\varphi)$ je rovna 1, proto je tento parametr pokládán rovněž konstantní, a sice $\alpha_k \equiv \alpha \equiv 1$. Výpočetní kroky probíhají tak, aby se všechny iterace $(\mathbf{x},\mathbf{y},\mathbf{s})$ nacházely uvnitř okolí $\mathscr{N}_2(\varphi)$ a míra duality μ_k konvergovala k nule konstantní rychlostí σ.

[1] někdy nazývané Mizuno-Todd-Ye prediktor-korektor metody, aby se zdůraznila odlišnost od algoritmů prediktor-korektor Mehrotrova typu

Speciálními volbami parametrů je možné z Algoritmu PD získat algoritmus metody sledování cesty s krátkým krokem v následující podobě:

Algoritmus CKK

Vstupy:

parametr okolí $\varphi = 0.4$,
centrující parametr $\sigma = 1 - \frac{0.4}{\sqrt{n}}$,
startovací bod $\left(\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0\right) \in \mathcal{N}_2(\varphi)$.

begin

for $k = 0, 1, 2, \ldots$

Pro $\mu_k = \frac{\left(\mathbf{x}^k\right)^{\mathrm{T}} \mathbf{s}^k}{n}$ najdi řešení soustavy

$$\begin{pmatrix} \mathbf{0} & \mathbf{A}^{\mathrm{T}} & \mathbf{I} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{S}^k & \mathbf{0} & \mathbf{X}^k \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x}^k \\ \Delta\mathbf{y}^k \\ \Delta\mathbf{s}^k \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ -\mathbf{X}^k\mathbf{S}^k\mathbf{e} + \sigma\mu_k\mathbf{e} \end{pmatrix}. \tag{2.3}$$

Polož $\left(\mathbf{x}^{k+1}, \mathbf{y}^{k+1}, \mathbf{s}^{k+1}\right) = \left(\mathbf{x}^k + \Delta\mathbf{x}^k, \mathbf{y}^k + \Delta\mathbf{y}^k, \mathbf{s}^k + \Delta\mathbf{s}^k\right)$.

end (for)

end

Analýza Algoritmu CKK ukazuje, že všechny generované iterace leží uvnitř okolí $\mathcal{N}_2(\varphi)$, dále také globální konvergenci a jeho polynomiální složitost. Platí-li požadavek, že se všechny iterace nachází v okolí $\mathcal{N}_2(\varphi)$, je možné ukázat globální konvergenci a polynomiální složitost, což bude provedeno hned po následujícím lemmatu[2], z něhož vyplývá lineární konvergence. A to při zavedení označení zachycujícího závislost veličin na délce kroku:

$$\begin{aligned} (\mathbf{x}(\alpha), \mathbf{y}(\alpha), \mathbf{s}(\alpha)) &= (\mathbf{x} + \alpha_k\Delta\mathbf{x}, \mathbf{y} + \alpha_k\Delta\mathbf{y}, \mathbf{s} + \alpha_k\Delta\mathbf{s}), \\ \sigma(\alpha) &= \frac{\mathbf{x}(\alpha)^{\mathrm{T}}\mathbf{s}(\alpha)}{n} \end{aligned} \tag{2.4}$$

Posléze bude ukázáno, že je možné používat $\alpha = 1$.

Lemma 2.1. Nechť krok $(\Delta\mathbf{x}, \Delta\mathbf{y}, \Delta\mathbf{s})$ je definován soustavou (1.12).

Potom

$$\Delta\mathbf{x}^{\mathrm{T}}\Delta\mathbf{s} = 0 \tag{2.5}$$

a

$$\mu(\alpha) = (1 - \alpha(1 - \sigma))\mu. \tag{2.6}$$

[2]Obecný tvar je použit proto, aby lemma bylo možné aplikovat na všechny druhy algoritmů, jež užívají k výpočtu směru $(\Delta\mathbf{x}, \Delta\mathbf{y}, \Delta\mathbf{s})$ soustavu (1.13).

Důkaz. viz [7], příp.[2]. □

Z tohoto lemmatu plyne globální lineární konvergence uvedeného Algoritmu CKK, neboť při speciální volbě parametrů $\sigma_k = \sigma = 1 - \frac{0.4}{\sqrt{n}}$ a $\alpha_k = 1$, platí

$$\mu_{k+1} = \sigma\mu_k = \left(1 - \frac{0.4}{\sqrt{n}}\right)\mu_k, \qquad k = 0, 1, \ldots \tag{2.7}$$

Polynomiální složitost Algoritmu CKK je prezentována

Věta 2.1. Nechť $\varepsilon > 0$. Předpokládejme, že startovací bod $\left(\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0\right) \in \mathcal{N}_2(0.4)$ v Algoritmu CKK splňuje

$$\mu_0 \leq \frac{1}{\varepsilon^\kappa} \tag{2.8}$$

pro libovolnou kladnou konstantu κ.

Potom existuje index $K = O\left(\sqrt{n}\, log\, \frac{1}{\varepsilon}\right)$ takový, že

$$\mu_k \leq \varepsilon \qquad pro\ \forall k \geq K.$$

Důkaz. viz [2], příp. [7].

□

Lemma 2.1 ukazuje, že součin $\Delta\mathbf{x}^{\mathrm{T}}\Delta\mathbf{s} = \sum_{i=1}^{n} \Delta x_i \Delta s_i$ je roven nule, což ovšem neznamená, že také jednotlivé součiny $\Delta x_i \Delta s_i$, $i = 1, ..., n$ jsou nulové. Jistou mez pro normu vektoru těchto součinů stanovuje následující lemma:

Lemma 2.2. Jestliže $(\mathbf{x}, \mathbf{y}, \mathbf{s}) \in \mathcal{N}_2(\varphi)$, potom

$$\|\Delta\mathbf{X}\Delta\mathbf{S}\mathbf{e}\| \leq \frac{\varphi^2 + n(1-\sigma)^2}{2^{\frac{3}{2}}(1-\varphi)}\mu.$$

Důkaz. viz [2], příp. [7]. □

Vzdálenost bodu $(\mathbf{x}(\alpha), \mathbf{y}(\alpha), \mathbf{s}(\alpha))$, nacházejícího se v okolí $\mathcal{N}_2(\varphi)$, od centrální cesty při použití Euklidovské normy určuje následující lemma, které je jednoduchým důsledkem Lemma 2.2.

Lemma 2.3. Jestliže $(\mathbf{x}, \mathbf{y}, \mathbf{s}) \in \mathcal{N}_2(\varphi)$, potom platí

$$\begin{aligned} \|\mathbf{X}(\alpha)\mathbf{S}(\alpha)\mathbf{e} - \mu(\alpha)\mathbf{e}\| &\leq |1-\alpha|\,\|\mathbf{X}\mathbf{S}\mathbf{e} - \mu\mathbf{e}\| + \alpha^2\,\|\Delta\mathbf{X}\Delta\mathbf{S}\mathbf{e}\| \\ &\leq |1-\alpha|\,\varphi\mu + \alpha^2\,\frac{\varphi^2 + n(1-\sigma)^2}{2^{\frac{3}{2}}(1-\varphi)}\mu. \end{aligned}$$

Důkaz. viz [2], příp. [7]. □

Nyní bude uvedena věta definující vztah mezi parametry φ a σ , která ukazuje, že i v případě plného kroku ($\alpha = 1$), se nová iterace nachází v okolí $\mathcal{N}_2(\varphi)$.

Věta 2.2. Nechť pro parametry $\varphi \in (0,1)$ a $\sigma \in (0,1)$ platí

$$\frac{\varphi^2 + n(1-\sigma)^2}{2^{\frac{3}{2}}(1-\varphi)}\mu \leq \sigma\varphi. \tag{2.9}$$

Potom, jestliže $(\mathbf{x},\mathbf{y},\mathbf{s}) \in \mathcal{N}_2(\varphi)$, tak

$$(\mathbf{x}(\alpha),\mathbf{y}(\alpha),\mathbf{s}(\alpha)) \in \mathcal{N}_2(\varphi) \qquad pro\ \forall\alpha \in [0,1].$$

Důkaz. viz [7], příp. [2]. □

Ověření platnosti Algoritmu CKK je úplné, neboť pro zvolené speciální hodnoty parametrů $\varphi = 0.4$ a $\sigma = 1 - \frac{0.4}{\sqrt{n}}$ je splněna podmínka (2.9) pro každé $n \geq 1$ (důkaz je možné nalézt v [7]).

Metody sledování cesty s krátkým krokem je dobře teoreticky zvládnutou metodou, avšak kvůli použití „úzkého“ okolí $\mathcal{N}_2(\varphi)$ je značně omezen postup k optimálnímu řešení. Při použití v praxi to velmi často znamená značnou časovou náročnost výpočtu, proto se touto metodou práce blíže nezabývá.

2.2 Metody sledování (centrální) cesty s dlouhým krokem

Metody sledování cesty s dlouhým krokem užívají „široké“ okolí $\mathcal{N}_{-\infty}(\gamma)$, které pro hodnoty γ blízké nule zaujímá většinu striktně přípustné oblasti $\mathcal{F}^0$ a umožňuje rychlý postup směrem k optimálnímu řešení.

Algoritmus metody vychází z bodu $\left(\mathbf{x}^0,\mathbf{y}^0,\mathbf{s}^0\right) \in \mathcal{N}_{-\infty}(\gamma)$ a generuje posloupnost iterací nacházejících se opět v tomto okolí. V každé iteraci se volí centrující parametr σ_k tak, aby jeho hodnota ležela mezi pevně stanovenými konstantami σ_{min} a σ_{max}, pro něž platí $0 < \sigma_{min} < \sigma_{max} < 1$. Dolní mez centrujícího parametru σ_{min} zabezpečuje, že směry iterací jsou odvedeny od hranice okolí $\mathcal{N}_{-\infty}(\gamma)$ do jeho vnitřku. Z toho vyplývá, že malé hodnoty parametru délky kroku α v tomto směru vylepšují centralitu. Velké hodnoty α však mohou mít za následek opět pohyb za hranici okolí, což si vynucuje jeho zkrácení. Dále je určen směr jednotlivých iterací, a to řešením soustavy (1.13). Délka kroku se vybírá největší možná, zachovávající další iteraci v okolí $\mathcal{N}_{-\infty}(\gamma)$.

Algoritmus metody (získaný úpravou Algoritmu PD) za předpokladu, že startovací bod je striktně přípustný vypadá následovně:

Algoritmus CDK

Vstupy:

parametr okolí $\gamma \in (0,1)$,
*meze centrujícího param*etru σ_{min}, σ_{max}: $0 < \sigma_{min} < \sigma_{max} < 1$,
startovací bod $\left(\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0\right) \in \mathcal{N}_{-\infty}(\gamma)$.

begin

for $k = 0, 1, 2, ...$

Zvol $\sigma_k \in [\sigma_{min}, \sigma_{max}]$.

Pro $\mu_k = \frac{\left(\mathbf{x}^k\right)^{\mathrm{T}} \mathbf{s}^k}{n}$ najdi řešení systému

$$\begin{pmatrix} \mathbf{0} & \mathbf{A}^{\mathrm{T}} & \mathbf{I} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{S}^k & \mathbf{0} & \mathbf{X}^k \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x}^k \\ \Delta\mathbf{y}^k \\ \Delta\mathbf{s}^k \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ -\mathbf{X}^k\mathbf{S}^k\mathbf{e} + \sigma_k\mu_k\mathbf{e} \end{pmatrix}. \tag{2.10}$$

Vypočítej maximální délku kroku $\alpha_k \in [0,1]$ splňující

$$\left(\mathbf{x}^k + \alpha_k\Delta\mathbf{x}^k, \mathbf{y}^k + \alpha_k\Delta\mathbf{y}^k, \mathbf{s}^k + \alpha_k\Delta\mathbf{s}^k\right) \in \mathcal{N}_{-\infty}(\gamma).$$

Polož $\left(\mathbf{x}^{k+1}, \mathbf{y}^{k+1}, \mathbf{s}^{k+1}\right) = \left(\mathbf{x}^k + \alpha_k\Delta\mathbf{x}^k, \mathbf{y}^k + \alpha_k\Delta\mathbf{y}^k, \mathbf{s}^k + \alpha_k\Delta\mathbf{s}^k\right)$.

end (for)

end

Následující věty poskytují odpověď na otázku, jak volit minimální délku kroku α v závislosti na vstupních patrametrech, jež zaručuje, že nedojde k opuštění okolí $\mathcal{N}_{-\infty}(\gamma)$. Dále také uvádí odhad redukce míry duality μ v každé iteraci a polynomiální složitost. Nejprve je uvedeno lemma stanovující mez pro vektor součinů $\Delta x_i \Delta s_i$, $i = 1, ..., n$:

Lemma 2.4. Jestliže $(\mathbf{x}, \mathbf{y}, \mathbf{s}) \in \mathcal{N}_{-\infty}(\gamma)$, potom

$$\|\Delta\mathbf{X}\Delta\mathbf{S}\mathbf{e}\| \leq 2^{-\frac{3}{2}} \left(1 + \frac{1}{\gamma}\right) n\mu.$$

Důkaz. viz [2], příp. [7].

Na základě výsledku Lemmatu 2.4 je možné uvést následující větu věnující se odhadu redukce míry duality μ v každé iteraci a dolní mezi pro délku kroku α, z níž jako bezprostřední důsledek vyplývá globální konvergence. □

Věta 2.3. Nechť jsou dány parametry γ, σ_{min} a σ_{max} z Algoritmu CDK.

Potom existuje konstanta η nezávislá na n taková, že

$$\mu_{k+1} \leq \left(1 - \frac{\eta}{n}\right) \mu_k \qquad pro\ \forall k \geq 0.$$

Důkaz. viz [7], příp.[2] nebo [1]. □

O polynomiální složitosti Algoritmu CDK hovoří následující věta:

Věta 2.4. Nechť je dáno $\varepsilon > 0$ a $\gamma \in (0,1)$. Předpokládejme, že pro startovací bod $\left(\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0\right) \in \mathscr{N}_{-\infty}(\gamma)$ z Algoritmu CDK platí

$$\mu_0 \leq \frac{1}{\varepsilon^\kappa}$$

pro nějakou konstantu $\kappa > 0$.

Potom existuje index $K = O\left(n\, log\, \frac{1}{\varepsilon}\right)$, takový, že

$$\mu_k \leq \varepsilon \qquad pro\ \forall k \geq K.$$

Důkaz. viz [1]. □

„Nepřípustná“ varianta metody sledování cesty s dlouhým krokem, jak již bylo naznačeno, vychází z bodu $\left(\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0\right)$, pro který platí, že $\mathbf{x}^0 > \mathbf{0}$ a $\mathbf{s}^0 > \mathbf{0}$. Obecně jsou všechny iterace $\left(\mathbf{x}^k, \mathbf{y}^k, \mathbf{s}^k\right)$ generované algoritmem nepřípustné, ačkoliv jeho limitní bod je samozřemě přípustný a optimální. Tato situace vyžaduje úpravu definice okolí $\mathscr{N}_{-\infty}(\gamma)$, aby umožnila porušit podmínky přípustnosti, a okolí tak mohlo obsahovat i nepřípustné body a zároveň byl zaručen pokles reziduí $\mathbf{r}_b$ a $\mathbf{r}_c$ přibližně stejnou rychlostí jako míra duality μ. Nová definice $\mathscr{N}_{-\infty}$ je tedy:

$$\mathscr{N}_{-\infty}(\gamma, \delta) = \left\{ (\mathbf{x}, \mathbf{y}, \mathbf{s}) \mid \|(\mathbf{r}_b, \mathbf{r}_c)\| \leq \frac{\left\|\left(\mathbf{r}_b^0, \mathbf{r}_c^0\right)\right\|}{\mu_0} \delta\mu,\ (\mathbf{x}, \mathbf{s}) > \mathbf{0},\ x_i s_i \geq \gamma\mu \text{ pro } i = 1, 2, \ldots, n \right\}, \tag{2.11}$$

kde $\gamma \in (0,1)$ a $\delta \geq 1$ jsou dané parametry, $\left(\mathbf{r}_b^0, \mathbf{r}_c^0\right)$ a μ_0 jsou vypočítány ze startovacího bodu $\left(\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0\right)$ a $(\mathbf{r}_b, \mathbf{r}_c)$ a μ jsou z přílušného bodu bodu $(\mathbf{x}, \mathbf{y}, \mathbf{s})$.

K určení Newtonova směru je využita soustava (1.14). Co se týče parametrů, je podmínka $0 < \sigma_{min} < \sigma_{max} < 1$ pro možnou volbu centrujícího parametru σ_k z algoritmu CDK nahrazena nerovností $0 < \sigma_{min} < \sigma_{max} \leq 0.5$. Dále je požadováno, aby délka kroku α volená v každé iteraci z intervalu $[0,1]$ byla největší možná, což zde znamená, že zachovává iteraci v okolí $\mathscr{N}_{-\infty}(\gamma, \delta)$ a zároveň splňuje Armijovu podmínku tvaru:

$$\mu(\alpha) \leq (1-0.01\alpha)\,\mu, \tag{2.12}$$

kde $\mu(\alpha) = \frac{(\mathbf{x}+\alpha\Delta\mathbf{x})^{\mathrm{T}}(\mathbf{s}+\alpha\Delta\mathbf{s})}{n}$.

Armiova podmínka zajišťuje alespoň nepatrný pokles míry duality μ v každé iteraci. Ve skutečnosti je možné délku kroku volit i menší (ovšem ne příliš), pokud platí výše uvedené podmínky (více informací je možné naleznout v [2]).

Celý algoritmus „nepřípustné“ metody sledování cesty s dlouhým krokem vypadá následovně:

Algoritmus NCDK

Vstupy:

parametry okolí $\gamma \in (0,1)$, $\delta \geq 1$,

*meze centrujícího param*etru σ_{min},σ_{max}: $0 < \sigma_{min} < \sigma_{max} \leq 0.5$,

startovací bod $\left(\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0\right)$: $\mathbf{x}^0 > \mathbf{0}$, $\mathbf{s}^0 > \mathbf{0}$.

begin

for $k = 0,1,2,...$

Zvol $\sigma_k \in [\sigma_{min}, \sigma_{max}]$.

Pro $\mathbf{r}_c^k = \mathbf{A}^{\mathrm{T}}\mathbf{y}^k + \mathbf{s}^k - \mathbf{c}$, $\mathbf{r}_b^k = \mathbf{A}\mathbf{x}^k - \mathbf{b}$ a $\mu_k = \frac{\left(\mathbf{x}^k\right)^{\mathrm{T}}\mathbf{s}^k}{n}$ najdi řešení systému

$$\begin{pmatrix} \mathbf{0} & \mathbf{A}^{\mathrm{T}} & \mathbf{I} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{S}^k & \mathbf{0} & \mathbf{X}^k \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x}^k \\ \Delta\mathbf{y}^k \\ \Delta\mathbf{s}^k \end{pmatrix} = \begin{pmatrix} -\mathbf{r}_c^k \\ -\mathbf{r}_b^k \\ -\mathbf{X}^k\mathbf{S}^k\mathbf{e} + \sigma_k\mu_k\mathbf{e} \end{pmatrix}. \tag{2.13}$$

Vypočítej maximální délku kroku $\alpha_k \in [0,1]$ splňující

$$\left(\mathbf{x}^k + \alpha_k\Delta\mathbf{x}^k, \mathbf{y}^k + \alpha_k\Delta\mathbf{y}^k, \mathbf{s}^k + \alpha_k\Delta\mathbf{s}^k\right) \in \mathscr{N}_{-\infty}(\gamma,\delta)$$

a Armiovu podmínku:

$$\mu_k(\alpha) \leq (1-0.01\alpha)\,\mu_k. \tag{2.14}$$

Polož $\left(\mathbf{x}^{k+1}, \mathbf{y}^{k+1}, \mathbf{s}^{k+1}\right) = \left(\mathbf{x}^k + \alpha_k\Delta\mathbf{x}^k, \mathbf{y}^k + \alpha_k\Delta\mathbf{y}^k, \mathbf{s}^k + \alpha_k\Delta\mathbf{s}^k\right)$.

end (for)

end

Tento algoritmus funguje, i když je striktně přípustná množina $\mathscr{F}^0$ prázdná. V přípustném případě $\mathbf{r}_b^0 = \mathbf{0}$, $\mathbf{r}_b^0 = \mathbf{0}$, všechny iterace $\left(\mathbf{x}^k, \mathbf{y}^k, \mathbf{s}^k\right)$ jsou striktně přípustné a algoritmus se převádí na Algoritmus CDK.

V případě speciální volby startovacího bodu $\left(\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0\right) = (\zeta\mathbf{e}, \mathbf{0}, \zeta\mathbf{e})$, kde ζ je dostatečně velké, vykazuje Algoritmus NCDK kvadratickou složitost: bod splňující $\mu_k \leq \varepsilon$, kde ε je daná přesnost, je získán v $\mathrm{O}\left(n^2 \left|log\, \varepsilon\right|\right)$ iteracích.

Než bude uvedena věta popisující globální konvergenci Algoritmu NCDK je třeba definovat některé pojmy:

Definition 2.1. Nechť $\{\psi_k\}$ je posloupnost kladných čísel, která konverguje k nule, tj. $\lim\limits_{k\to\infty} \psi_k = 0$.

Řekneme, že posloupnost $\{\psi_k\}$ *konverguje Q-lineárně k nule*, jestliže existuje číslo $\rho \in (0,1)$ takové, že $\frac{\psi_{k+1}}{\eta_k} \leq \rho$ pro $\forall k$ dostatečně velké.

Řekneme, že posloupnost $\{\psi_k\}$ *konverguje R-lineárně k nule*, jestliže je dominována posloupností, jež konverguje k nule Q-lineárně, tj. jestliže existuje posloupnost $\{\widehat{\eta}_k\}$ taková, že $0 \leq \psi_k \leq \widehat{\psi}_k$ pro $\forall k$, kde $\{\widehat{\psi}_k\}$ konverguje Q-lineárně.

Věta charakterizující globální konvergenci Algoritmu NCDK zní takto:

Věta 2.5. Posloupnost $\{\mu_k\}$ generovaná Algoritmem NCDK konverguje Q-lineárně k nule a posloupnost norem reziduí $\left\{\left\|\left(\mathbf{r}_b^k, \mathbf{r}_c^k\right)\right\|\right\}$ konverguje R-lineárně k nule.

Důkaz. viz [2]. □

Pro startovací bod $\left(\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0\right) = (\zeta\mathbf{e}, \mathbf{0}, \zeta\mathbf{e})$ je možné formulovat tvrzení o složitosti algoritmu, předtím ale je třeba uvést lemma definující ζ:

Lemma 2.5. Předpokládejme, že pro startovací bod $\left(\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0\right) = (\zeta\mathbf{e}, \mathbf{0}, \zeta\mathbf{e})$ platí

$$\left\|\left(\mathbf{x}^*, \mathbf{s}^*\right)\right\|_\infty \leq \zeta$$

pro nějaké primárně-duální řešení $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{s}^*)$.

Potom pro libovolnou iteraci $\left(\mathbf{x}^k, \mathbf{y}^k, \mathbf{s}^k\right)$ platí

$$\zeta\nu_k \left\|\left(\mathbf{x}^k, \mathbf{s}^k\right)\right\|_1 \leq 4\delta n\mu_k,$$

kde $\nu_k = \prod\limits_{j=0}^{k-1}\left(1-\alpha_j\right)$, přičemž $\nu_0 = 1$.

Důkaz. viz [2].

□

Věta 2.6. Nechť je dáno $\varepsilon > 0$. Předpokládejme, že startovací bod $\left(\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0\right) = (\zeta\mathbf{e}, \mathbf{0}, \zeta\mathbf{e})$, kde ζ je definováno v Lemma 2.5. Dále předpokládejme, ζ že hodnota pro náš konkrétní případ lineárního programování splňuje

$$\zeta^2 \leq \frac{C}{\varepsilon^\kappa}$$

pro libovolné kladné konstanty C a κ.

Potom existuje index $K = O\left(n^2 \left|log\,\varepsilon\right|\right)$ takový, že pro posloupost iterací $\left\{\left(\mathbf{x}^k, \mathbf{y}^k, \mathbf{s}^k\right)\right\}$ generovanou Algoritmem NCDK platí

$$\mu_k \leq \varepsilon \qquad pro\ \forall k \geq K.$$

Důkaz. viz [2]. □

Kapitola 3

Algoritmy Mehrotrova typu

3.1 Mehrotrův algoritmus

Jedním z nejužívanějších algoritmů v softwarech využívajících metod vnitřních bodů pro řešení úloh lineárního programování je praktický primárně-duální algoritmus typu prediktor-korektor[1] navržený Mehrotrem. Mehrotrův algoritmus patří mezi „nepřípustné“ metody vnitřních bodů, tedy umožňuje použít i nepřípustný startovací bod, který však musí splňovat podmínky kladnosti, tj. $\mathbf{x}^0 > \mathbf{0}$ a $\mathbf{s}^0 > \mathbf{0}$.

Mehrotrův algoritmus vylepšuje základní Newtonův směr určovaný výpočtem soustavy (1.14) o korekční složku tím, že při aproximaci centrální cesty využívá kvadratickou informaci o ní.

Dovoluje také volit centrující parametr σ adaptivně v každé iteraci. Kombinací již známých poznatků se Mehrotrovi podařilo vytvořit důmyslnou heurestiku pro volbu centrujícího parametru σ, délky kroku α a startovacího bodu $\left(\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0\right)$, jež při hledání optimálního primárně-duálního řešení úloh lineárního programování pracuje velmi efektivně.

3.1.1 Popis Mehrotrova algoritmu

Mehrotrův algoritmus generuje posloupnost nepřípustných iterací $\left(\mathbf{x}^k, \mathbf{y}^k, \mathbf{s}^k\right)$, pro jejichž složky platí, že $\mathbf{x}^k > \mathbf{0}$ a $\mathbf{s}^k > \mathbf{0}$. Hledaný „Mehrotrův“ směr, vedoucí k další iteraci, se v každé iteraci stanovuje kombinací tří složek:

- *„predikční“ směr*, což je základní Newtonův směr (affine scaling směr) pro funkci $F(\mathbf{x}, \mathbf{y}, \mathbf{s})$ definovanou v (1.4),

[1] Algoritmy typu prediktor-korektor jsou konstruovány tak, že zlepšení centrality a redukce míry duality je pro každou iteraci dosahováno ve dvou samostatných krocích:

- predikční krok - slouží k redukci míry duality,
- korekční krok - slouží k vylepšení centrality.

- *centrující složka*, odvislá od adaptivního výběru centrujícího parametru σ,
- *„korekční“ směr*, který se pokouší kompenzovat nelinearitu v affine scaling směru.

Následuje konkrétní postup určení „Mehrotrova“ směru, je-li dán bod $(\mathbf{x},\mathbf{y},\mathbf{s})$, pro nějž platí, že $\mathbf{x}>\mathbf{0}$ a $\mathbf{s}>\mathbf{0}$. Nejprve je stanoven affine scaling směr $(\Delta\widetilde{\mathbf{x}},\Delta\widetilde{\mathbf{y}},\Delta\widetilde{\mathbf{s}})$ vyřešením soustavy

$$\begin{pmatrix} \mathbf{0} & \mathbf{A}^{\mathrm{T}} & \mathbf{I} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{S} & \mathbf{0} & \mathbf{X} \end{pmatrix} \begin{pmatrix} \Delta\widetilde{\mathbf{x}} \\ \Delta\widetilde{\mathbf{y}} \\ \Delta\widetilde{\mathbf{s}} \end{pmatrix} = \begin{pmatrix} -\mathbf{r}_c \\ -\mathbf{r}_b \\ -\mathbf{XSe} \end{pmatrix}, \tag{3.1}$$

kde $\mathbf{r}_b$ a $\mathbf{r}_c$ jsou rezidua definovaná stejně jako výše vztahy

$$\begin{aligned} \mathbf{r}_b &= \mathbf{A}\mathbf{x}-\mathbf{b}, \\ \mathbf{r}_c &= \mathbf{A}^{\mathrm{T}}\mathbf{y}+\mathbf{s}-\mathbf{c}. \end{aligned}$$

V nalezeném směru jsou určovány maximální délky kroků $\widetilde{\alpha}^{pri}\in(0,1]$ (krok pro primární proměnné) a $\widetilde{\alpha}^{dual}\in(0,1]$ (krok pro duální proměnné), které ještě neporušují podmínky nezápornosti $\mathbf{x}\geq\mathbf{0}$, $\mathbf{s}\geq\mathbf{0}$. Jejich hodnoty se hledají samostatně na základě vzorců

$$\widetilde{\alpha}^{pri} = min\left(1,\ \min_{i:\Delta\widetilde{x}_i<0} -\frac{x_i}{\Delta\widetilde{x}_i}\right), \tag{3.2}$$

$$\widetilde{\alpha}^{dual} = min\left(1,\ \min_{i:\Delta\widetilde{s}_i<0} -\frac{s_i}{\Delta\widetilde{s}_i}\right). \tag{3.3}$$

Míru duality affine scaling směru vyjadřuje $\widetilde{\mu}$ definované jako hypotetická hodnota μ, získaná plným krokem k hranici nezáporného orthantu:

$$\widetilde{\mu}=\frac{\left(\mathbf{x}+\widetilde{\alpha}^{pri}\Delta\widetilde{\mathbf{x}}\right)^{\mathrm{T}}\left(\mathbf{s}+\widetilde{\alpha}^{dual}\Delta\widetilde{\mathbf{s}}\right)}{n}.$$

Poté je stanovena hodnota centrujícího parametru na základě výrazu

$$\sigma=\left(\frac{\widetilde{\mu}}{\mu}\right)^3.$$

V případě, že $\widetilde{\mu}\ll\mu$, je affine scaling směr dobrým směrem, zabezpečujícím významnou redukci míry duality μ bez porušení podmínek kladnosti $\mathbf{x}>0$, $\mathbf{s}>0$, proto σ vypočítané z tohoto vzorce bude malé (blízké nule), což odpovídá pouze mírnému centrování. A naopak, je-li $\widetilde{\mu}$ přibližně stejné hodnoty jako μ, umožňuje affine scaling směr pouze malý krok, aniž by byly porušeny podmínky kladnosti. V tomto případě bude σ blízké 1, které zajišťuje mnohem větší centrování a poskytuje možnost značně redukovat μ v další iteraci.

K výpočtu centrující složky je třeba řešit soustavu (3.1), v níž je pravá strana nahrazena vektorem $(\mathbf{0},\mathbf{0},\sigma\mu\mathbf{e})^{\mathrm{T}}$, tj.

$$\begin{pmatrix} \mathbf{0} & \mathbf{A}^{\mathrm{T}} & \mathbf{I} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{S} & \mathbf{0} & \mathbf{X} \end{pmatrix} \begin{pmatrix} \Delta\widehat{\mathbf{x}} \\ \Delta\widehat{\mathbf{y}} \\ \Delta\widehat{\mathbf{s}} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \sigma\mu\mathbf{e} \end{pmatrix}.$$

Třetí složka „Mehrotrova“ směru - korekční směr - je počítána z téže soustavy, ovšem s tím, že pravá strana je substituována vektorem $\left(\mathbf{0},\mathbf{0},-\Delta\widetilde{\mathbf{X}}\Delta\widetilde{\mathbf{S}}\mathbf{e}\right)^{\mathrm{T}}$, tedy

$$\begin{pmatrix} \mathbf{0} & \mathbf{A}^{\mathrm{T}} & \mathbf{I} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{S} & \mathbf{0} & \mathbf{X} \end{pmatrix} \begin{pmatrix} \Delta\overline{\mathbf{x}} \\ \Delta\overline{\mathbf{y}} \\ \Delta\overline{\mathbf{s}} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ -\Delta\widetilde{\mathbf{X}}\Delta\widetilde{\mathbf{S}}\mathbf{e} \end{pmatrix},$$

kde

$$\begin{aligned} \Delta\widetilde{\mathbf{X}} &= diag\left(\Delta\widetilde{x}_1,\Delta\widetilde{x}_2,\ldots,\Delta\widetilde{x}_n\right), \\ \Delta\widetilde{\mathbf{S}} &= diag\left(\Delta\widetilde{s}_1,\Delta\widetilde{s}_2,\ldots,\Delta\widetilde{s}_n\right). \end{aligned}$$

Kompletní „Mehrotrův“ směr zahrnující „predikční“ směr, centrující složku i „korekční“ směr, je potom určen řešením jediné soustavy[2]:

$$\begin{pmatrix} \mathbf{0} & \mathbf{A}^{\mathrm{T}} & \mathbf{I} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{S} & \mathbf{0} & \mathbf{X} \end{pmatrix} \begin{pmatrix} \triangle\mathbf{x} \\ \triangle\mathbf{y} \\ \triangle\mathbf{s} \end{pmatrix} = \begin{pmatrix} -\mathbf{r}_c \\ -\mathbf{r}_b \\ -\mathbf{XSe}-\Delta\widetilde{\mathbf{X}}\Delta\widetilde{\mathbf{S}}\mathbf{e}+\sigma\mu\mathbf{e} \end{pmatrix}. \tag{3.4}$$

Její matice se shoduje s maticí soustav jednotlivých složek směru a pravá strana je součet jejich pravých stran.

V nalezeném „Mehrotrově“ směru jsou hledány největší možné délky primárního a duálního kroku, tedy takové, pro něž nejsou porušeny podmínky kladnosti $\mathbf{x}>\mathbf{0}$, $\mathbf{s}>\mathbf{0}$ pomocí vztahů

$$\alpha^{pri} = min\left(1,\tau \min_{i:\Delta x_i<0} -\frac{x_i}{\Delta x_i}\right), \tag{3.5}$$

$$\alpha^{dual} = min\left(1,\tau \min_{i:\Delta s_i<0} -\frac{s_i}{\Delta s_i}\right), \tag{3.6}$$

kde parametr $\tau\in[0.9,1.0)$.

Nastíněný postup je shrnut do Mehrotrova algoritmu, který probíhá ve dvou krocích:

[2]Výpočet jednotlivých složek „Mehrotrova“ směru je činěn pomocí stejné matice koeficientů, proto je možné je sloučit do jediného směru, kterému odpovídá jediná soustava.

- Nejprve se v tzv. *predikčním kroku* (zkráceně *prediktoru*) počítá affine scaling směr $(\Delta\widetilde{\mathbf{x}}, \Delta\widetilde{\mathbf{y}}, \Delta\widetilde{\mathbf{s}})$, ve kterém se určují délky kroků $\widetilde{\alpha}^{pri} \in (0,1]$ a $\widetilde{\alpha}^{dual} \in (0,1]$, aby byly posléze využity k výpočtu míry duality affine scaling směru $\widetilde{\mu}$ potřebné pro odhad centrujícího parametru σ.

- Poté v tzv. *korekčním kroku* (zkráceně *korektoru*) jsou hodnoty $\widetilde{\mu}$ a σ použity k nalezení „Mehrotrova“ směru $(\Delta\mathbf{x}, \Delta\mathbf{y}, \Delta\mathbf{s})$, ve kterém stanoví délku kroků $\alpha^{pri} \in (0,1]$ a $\alpha^{dual} \in (0,1]$, aby se pomocí nich určila nová iterace.

3.1.2 Schéma Mehrotrova algoritmu

Algoritmus MA

Vstupy:

parametr $\tau \in [0.9, 1.0)$.

startovací bod $\left(\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0\right) : \mathbf{x}^0 > \mathbf{0}, \mathbf{s}^0 > \mathbf{0}$.

begin

for $k = 0, 1, 2, \ldots$

Predikční krok

Pro $\mathbf{r}_c^k = \mathbf{A}^{\mathrm{T}}\mathbf{y}^k + \mathbf{s}^k - \mathbf{c}$ a $\mathbf{r}_b^k = \mathbf{A}\mathbf{x}^k - \mathbf{b}$ najdi řešení systému

$$\begin{pmatrix} \mathbf{0} & \mathbf{A}^{\mathrm{T}} & \mathbf{I} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{S}^k & \mathbf{0} & \mathbf{X}^k \end{pmatrix} \begin{pmatrix} \triangle\widetilde{\mathbf{x}}^k \\ \triangle\widetilde{\mathbf{y}}^k \\ \triangle\widetilde{\mathbf{s}}^k \end{pmatrix} = \begin{pmatrix} -\mathbf{r}_c^k \\ -\mathbf{r}_b^k \\ -\mathbf{X}^k\mathbf{S}^k\mathbf{e} \end{pmatrix}.$$

Vypočítej přípustné kroky $\widetilde{\alpha}_k^{pri} \in (0,1]$, a $\widetilde{\alpha}_k^{dual} \in (0,1]$ použitím vztahů

$$\widetilde{\alpha}_k^{pri} = min\left(1, \min_{i:\Delta\widetilde{x}_i^k<0} -\frac{x_i^k}{\Delta\widetilde{x}_i^k}\right), \tag{3.7}$$

$$\widetilde{\alpha}_k^{dual} = min\left(1, \min_{i:\Delta\widetilde{s}_i^k<0} -\frac{s_i^k}{\Delta\widetilde{s}_i^k}\right). \tag{3.8}$$

Vypočítej míru duality řešení v prediktoru $\widetilde{\mu}_k = \frac{\left(\mathbf{x}^k + \widetilde{\alpha}_k^{pri}\triangle\widetilde{\mathbf{x}}^k\right)^{\mathrm{T}}\left(\mathbf{s}^k + \widetilde{\alpha}_k^{dual}\triangle\widetilde{\mathbf{s}}^k\right)}{n}$
a pro $\mu_k = \frac{\left(\mathbf{x}^k\right)^{\mathrm{T}}\mathbf{s}^k}{n}$ najdi centrující parametr

$$\sigma_k = \left(\frac{\widetilde{\mu}_k}{\mu_k}\right)^3.$$

end (Predikční krok)

Korekční krok

Pro $\mathbf{r}_c^k = \mathbf{A}^{\mathrm{T}}\mathbf{y}^k + \mathbf{s}^k - \mathbf{c}$ a $\mathbf{r}_b^k = \mathbf{A}\mathbf{x}^k - b$ najdi řešení systému

$$\begin{pmatrix} \mathbf{0} & \mathbf{A}^{\mathrm{T}} & \mathbf{I} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{S}^k & \mathbf{0} & \mathbf{X}^k \end{pmatrix} \begin{pmatrix} \triangle\mathbf{x}^k \\ \triangle\mathbf{y}^k \\ \triangle\mathbf{s}^k \end{pmatrix} = \begin{pmatrix} -\mathbf{r}_c^k \\ -\mathbf{r}_b^k \\ -\mathbf{X}^k\mathbf{S}^k\mathbf{e} - \triangle\widetilde{\mathbf{X}}^k\triangle\widetilde{\mathbf{S}}^k\mathbf{e} + \sigma_k\mu_k\mathbf{e} \end{pmatrix}.$$

Vypočítej přípustné kroky $\alpha_k^{pri} \in (0,1]$ a $\alpha_k^{dual} \in (0,1]$ pomocí vztahů

$$\alpha_k^{pri} = min\left(1, \tau \min_{i:\Delta x_i^k<0} -\frac{x_i^k}{\Delta x_i^k}\right), \tag{3.9}$$

$$\alpha_k^{dual} = min\left(1, \tau \min_{i:\Delta s_i^k<0} -\frac{s_i^k}{\Delta s_i^k}\right). \tag{3.10}$$

Polož $\left(\mathbf{x}^{k+1}, \mathbf{y}^{k+1}, \mathbf{s}^{k+1}\right) = \left(\mathbf{x}^k + \alpha_k^{pri}\triangle\mathbf{x}^k, \mathbf{y}^k + \alpha_k^{dual}\triangle\mathbf{y}^k, \mathbf{s}^k + \alpha_k^{dual}\triangle\mathbf{s}^k\right)$.

end (Korekční krok)

end (for)

end

Na tomto místě je třeba poznamenat hned dvě věci: jednak, že uvedený Mehrotrův algoritmus je jen jednou z několika uváděných variant a jednak, že pro uvedený algoritmus není k dispozici žádná konvergenční teorie. Ve skutečnosti existují příklady, pro které algoritmus diverguje. Do algoritmu ale mohou být včleněny určité prvky umožňující dokázat konvergenci i složitost. Dále jsou uvedeny některé takovéto modifikace včetně informací o konvergenci.

3.2 Modifikace Mehrotrova algoritmu

Zejména možná divergence Mehrotrova algoritmu, ale i malé délky kroků v „korekčním“ kroku (je-li délka kroku v affine scaling směru poměrně velká) nebo čisté centrující kroky v „korekčním“ kroku (při velmi malé délce kroku v affine scaling směru) vedly k úpravám Mehrotrovy heuristiky. Změny se dle [12] a [13] projevují v adaptivní volbě některých parametrů a v jisté regulaci výpočtu v „korekčním“ kroku: v jistých případech se upravuje délka[3] kroku $\widetilde{\alpha}$ vypočtená v „predikčním“ kroku nebo se používá upravené soustavy pro výpočet směru. A také v závislosti na délce kroku α, vypočtené v „korekčním“ kroku, může být vyžadován přepočet směru i α tak, aby byla zaručena jeho jistá minimální délka.

[3] zde určené jako minimum kroků $\widetilde{\alpha}^{pri}$ a $\widetilde{\alpha}^{dual}$

Dále jsou uvedeny čtyři algoritmy modifikací Mehrotrova algoritmu spolu s větami charakterizující jejich teoretickou iterační složitost. Všechny jsou popsány ve variantě se startovacím bodem $\left(\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0\right)$, který je striktně přípustný. Proto i všechny ostatní iterace, které jsou jimi generovány, jsou striktně přípustné. Bližší informace o jednotlivých modifikacích popisovaných v této práci je možné nalézt v článcích [12] a [13] (první dvě modifikace v [12] a druhé dvě v pořadí v [13]).

3.2.1 Modifikace č. 1

Algoritmus MMA1

Vstupy:

parametr okolí $\gamma \in \left(0, \frac{1}{4}\right)$,
startovací bod $\left(\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0\right) \in \mathscr{N}_{-\infty}(\gamma)$.

begin

for $k = 0, 1, 2, \ldots$

Predikční krok

Najdi řešení systému

$$\begin{pmatrix} \mathbf{0} & \mathbf{A}^{\mathrm{T}} & \mathbf{I} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{S}^k & \mathbf{0} & \mathbf{X}^k \end{pmatrix} \begin{pmatrix} \Delta\widetilde{\mathbf{x}}^k \\ \Delta\widetilde{\mathbf{y}}^k \\ \Delta\widetilde{\mathbf{s}}^k \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ -\mathbf{X}^k\mathbf{S}^k\mathbf{e} \end{pmatrix}.$$

Vypočítej maximální délku kroku $\widetilde{\alpha}_k$ splňující

$$\left(\mathbf{x}^k + \widetilde{\alpha}_k\Delta\mathbf{x}^k, \mathbf{y}^k + \widetilde{\alpha}_k\Delta\mathbf{y}^k, \mathbf{s}^k + \widetilde{\alpha}_k\Delta\mathbf{s}^k\right) \in \mathscr{F}.$$

end (Predikční krok)

Korekční krok

If $\widetilde{\alpha}_k \geq 0.1$ **then**

pro $\iota_k = (1 - \widetilde{\alpha}_k)^3 \mu_k$, kde $\mu_k = \frac{\left(\mathbf{x}^k\right)^{\mathrm{T}}\mathbf{s}^k}{n}$, najdi řešení systému

$$\begin{pmatrix} \mathbf{0} & \mathbf{A}^{\mathrm{T}} & \mathbf{I} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{S}^k & \mathbf{0} & \mathbf{X}^k \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x}^k \\ \Delta\mathbf{y}^k \\ \Delta\mathbf{s}^k \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ -\mathbf{X}^k\mathbf{S}^k\mathbf{e} - \Delta\widetilde{\mathbf{X}}^k\Delta\widetilde{\mathbf{S}}^k\mathbf{e} + \iota_k\mathbf{e} \end{pmatrix}. \tag{3.11}$$

Vypočítej maximální délku kroku α_k splňující

$$\left(\mathbf{x}^k + \alpha_k\Delta\mathbf{x}^k, \mathbf{y}^k + \alpha_k\Delta\mathbf{y}^k, \mathbf{s}^k + \alpha_k\Delta\mathbf{s}^k\right) \in \mathscr{N}_{-\infty}(\gamma).$$

end

If $\widetilde{\alpha}_k < 0.1$ **or** $\alpha_k < \frac{\gamma^2}{2n^2}$ **then**

pro $\iota_k = \frac{\gamma}{1-\gamma}\mu_k$, kde $\mu_k = \frac{\left(\mathbf{x}^k\right)^{\mathrm{T}}\mathbf{s}^k}{n}$, najdi řešení systému

$$\begin{pmatrix} \mathbf{0} & \mathbf{A}^{\mathrm{T}} & \mathbf{I} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{S}^k & \mathbf{0} & \mathbf{X}^k \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x}^k \\ \Delta\mathbf{y}^k \\ \Delta\mathbf{s}^k \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ -\mathbf{X}^k\mathbf{S}^k\mathbf{e} - \Delta\widetilde{\mathbf{X}}^k\Delta\widetilde{\mathbf{S}}^k\mathbf{e} + \iota_k\mathbf{e} \end{pmatrix}. \tag{3.12}$$

Vypočítej maximální délklu kroku α_k splňující

$$\left(\mathbf{x}^k + \alpha_k\Delta\mathbf{x}^k, \mathbf{y}^k + \alpha_k\Delta\mathbf{y}^k, \mathbf{s}^k + \alpha_k\Delta\mathbf{s}^k\right) \in \mathscr{N}_{-\infty}(\gamma).$$

end (if)

Polož $\left(\mathbf{x}^{k+1}, \mathbf{y}^{k+1}, \mathbf{s}^{k+1}\right) = \left(\mathbf{x}^k + \alpha_k\Delta\mathbf{x}^k, \mathbf{y}^k + \alpha_k\Delta\mathbf{y}^k, \mathbf{s}^k + \alpha_k\Delta\mathbf{s}^k\right)$.

end (Korekční krok)

end (for)

end

Pro právě uvedený algoritmus je možné narozdíl od Mehrotrova algoritmu dokázat polynomiální složitost:

Věta 3.1. Algoritmus MMA1 končí nejvýše po $O\left(n^2 log\frac{\left(x^0\right)^{\mathrm{T}}s^0}{\varepsilon}\right)$ iteracích s řešením, pro které platí $\mathbf{x}^{\mathrm{T}}\mathbf{s} \leq \varepsilon$.

Důkaz. viz [12]. □

3.2.2 Modifikace č. 2

Teoretickou iterační složitost Algoritmu MMA1 je možné významně snížit záměnou každé ze soustav (3.14), (3.15) řešených v „korekčním“ kroku za soustavu

$$\begin{pmatrix} \mathbf{0} & \mathbf{A}^{\mathrm{T}} & \mathbf{I} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{S}^k & \mathbf{0} & \mathbf{X}^k \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x}^k \\ \Delta\mathbf{y}^k \\ \Delta\mathbf{s}^k \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ -\mathbf{X}^k\mathbf{S}^k\mathbf{e} - \widetilde{\alpha}_k\Delta\widetilde{\mathbf{X}}^k\Delta\widetilde{\mathbf{S}}^k\mathbf{e} + \iota_k\mathbf{e} \end{pmatrix}. \tag{3.13}$$

Tato změna algoritmu je založena na skutečnosti, že pokud je affine scaling směr dobrou volbou, jež implikuje velkou délku kroku $\widetilde{\alpha}$, potom by klasický korekční směr měl být dobrým směrem. Jestliže ale není dobrou volbou, potom je třeba v korekčním kroku větší opatrnosti, promítané právě do členu obsahujicího $\widetilde{\alpha}$.

Schéma algoritmu pak vypadá následovně:

Algoritmus MMA2

Vstupy:

parametr okolí $\gamma \in \left(0, \frac{1}{4}\right)$,

startovací bod $\left(\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0\right) \in \mathscr{N}_{-\infty}(\gamma)$.

begin

for $k = 0, 1, 2, \ldots$

Predikční krok

Najdi řešení systému

$$\begin{pmatrix} \mathbf{0} & \mathbf{A}^{\mathrm{T}} & \mathbf{I} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{S}^k & \mathbf{0} & \mathbf{X}^k \end{pmatrix} \begin{pmatrix} \Delta\widetilde{\mathbf{x}}^k \\ \Delta\widetilde{\mathbf{y}}^k \\ \Delta\widetilde{\mathbf{s}}^k \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ -\mathbf{X}^k\mathbf{S}^k\mathbf{e} \end{pmatrix}.$$

Vypočítej maximální délku kroku $\widetilde{\alpha}_k$ splňující

$$\left(\mathbf{x}^k + \widetilde{\alpha}_k\Delta\mathbf{x}^k, \mathbf{y}^k + \widetilde{\alpha}_k\Delta\mathbf{y}^k, \mathbf{s}^k + \widetilde{\alpha}_k\Delta\mathbf{s}^k\right) \in \mathscr{F}.$$

end (Predikční krok)

Korekční krok

If $\widetilde{\alpha}_k \geq 0.1$ **then**

pro $\iota_k = (1 - \widetilde{\alpha}_k)^3 \mu_k$, kde $\mu_k = \frac{\left(\mathbf{x}^k\right)^{\mathrm{T}}\mathbf{s}^k}{n}$, najdi řešení systému

$$\begin{pmatrix} \mathbf{0} & \mathbf{A}^{\mathrm{T}} & \mathbf{I} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{S}^k & \mathbf{0} & \mathbf{X}^k \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x}^k \\ \Delta\mathbf{y}^k \\ \Delta\mathbf{s}^k \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ -\mathbf{X}^k\mathbf{S}^k\mathbf{e} - \widetilde{\alpha}_k\Delta\widetilde{\mathbf{X}}^k\Delta\widetilde{\mathbf{S}}^k\mathbf{e} + \iota_k\mathbf{e} \end{pmatrix}. \tag{3.14}$$

Vypočítej maximální délku kroku α_k splňující

$$\left(\mathbf{x}^k + \alpha_k\Delta\mathbf{x}^k, \mathbf{y}^k + \alpha_k\Delta\mathbf{y}^k, \mathbf{s}^k + \alpha_k\Delta\mathbf{s}^k\right) \in \mathscr{N}_{-\infty}(\gamma).$$

end (if)

If $\widetilde{\alpha}_k < 0.1$ **or** $\alpha_k < \frac{\gamma^2}{2n^2}$ **then**

pro $\iota_k = \frac{\gamma}{1-\gamma}\mu_k$, kde $\mu_k = \frac{\left(\mathbf{x}^k\right)^{\mathrm{T}}\mathbf{s}^k}{n}$, najdi řešení systému

$$\begin{pmatrix} \mathbf{0} & \mathbf{A}^{\mathrm{T}} & \mathbf{I} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{S}^k & \mathbf{0} & \mathbf{X}^k \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x}^k \\ \Delta\mathbf{y}^k \\ \Delta\mathbf{s}^k \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ -\mathbf{X}^k\mathbf{S}^k\mathbf{e} - \widetilde{\alpha}_k\Delta\widetilde{\mathbf{X}}^k\Delta\widetilde{\mathbf{S}}^k\mathbf{e} + \iota_k\mathbf{e} \end{pmatrix}. \tag{3.15}$$

Vypočítej maximální délklu kroku α_k splňující

$$\left(\mathbf{x}^k + \alpha_k\Delta\mathbf{x}^k, \mathbf{y}^k + \alpha_k\Delta\mathbf{y}^k, \mathbf{s}^k + \alpha_k\Delta\mathbf{s}^k\right) \in \mathscr{N}_{-\infty}(\gamma).$$

end (if)

Polož $\left(\mathbf{x}^{k+1}, \mathbf{y}^{k+1}, \mathbf{s}^{k+1}\right) = \left(\mathbf{x}^k + \alpha_k\Delta\mathbf{x}^k, \mathbf{y}^k + \alpha_k\Delta\mathbf{y}^k, \mathbf{s}^k + \alpha_k\Delta\mathbf{s}^k\right)$.

end (Korekční krok)

end (for)

end

O složitosti Algoritmu MMA2 hovoří následující věta:

Věta 3.2. Algoritmus MMA2 končí nejvýše po $O\left(n\,log\frac{n}{\varepsilon}\right)$ iteracích s řešením, pro které platí $\mathbf{x}^{\mathrm{T}}\mathbf{s} \leq \varepsilon$.

Důkaz. viz [12]. □

3.2.3 Modifikace č. 3

Pro parametr γ platí, že menší hodnota indikuje větší okolí, což v případě vztahu

$$\iota = \frac{\gamma}{1-\gamma}\mu \tag{3.16}$$

znamená poměrně agresivní strategii postupu k řešení, při níž ale hrozí, že se iterace dostanou „velmi brzy“ „velmi blízko“ hranice s následným poměrně pomalým postupem k optimálnímu řešení. Použití vztahu

$$\iota = \frac{\beta}{1-\beta}\mu, \tag{3.17}$$

kde $\beta \in \left[\gamma, \frac{1}{4}\right)$, ovšem umožňuje pracovat nezávisle na velikosti okolí, čímž taktiku určenou vzorcem (3.16) poněkud zmírňuje.

Algoritmus modifikace Mehrotrova algoritmu užívající vztah vypadá následovně:

Algoritmus MMA3

Vstupy:

parametr okolí $\gamma \in \left(0, \frac{1}{4}\right)$,
parametr záruky $\beta \in \left[\gamma, \frac{1}{4}\right)$,
startovací bod $\left(\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0\right) \in \mathscr{N}_{-\infty}(\gamma)$.

begin

for $k = 0, 1, 2, \ldots$

Predikční krok

Najdi řešení systému

$$\begin{pmatrix} \mathbf{0} & \mathbf{A}^{\mathrm{T}} & \mathbf{I} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{S}^k & \mathbf{0} & \mathbf{X}^k \end{pmatrix} \begin{pmatrix} \Delta\widetilde{\mathbf{x}}^k \\ \Delta\widetilde{\mathbf{y}}^k \\ \Delta\widetilde{\mathbf{s}}^k \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ -\mathbf{X}^k\mathbf{S}^k\mathbf{e} \end{pmatrix}.$$

Vypočítej maximální délku kroku $\widetilde{\alpha}_k$ splňující

$$\left(\mathbf{x}^k + \widetilde{\alpha}_k\Delta\mathbf{x}^k, \mathbf{y}^k + \widetilde{\alpha}_k\Delta\mathbf{y}^k, \mathbf{s}^k + \widetilde{\alpha}_k\Delta\mathbf{s}^k\right) \in \mathscr{F}.$$

end (Predikční krok)

Korekční krok

Spočítej $\widehat{\alpha}_k = 1 - \left(\frac{2\gamma t_k}{1-\gamma}\right)^{\frac{1}{3}}$, kde $t_k = \underset{i:\Delta\widetilde{x}_i^k\Delta\widetilde{s}_i^k>0}{max} \left\{\frac{\Delta\widetilde{x}_i^k\Delta\widetilde{s}_i^k}{x_i^k s_i^k}\right\}$.

If $\widetilde{\alpha}_k > \widehat{\alpha}_k$ **then**

polož $\widetilde{\alpha}_k = \widehat{\alpha}_k$.

end (if)

Pro $\iota_k = (1-\widetilde{\alpha}_k)^3\,\mu_k$, kde $\mu_k = \frac{\left(\mathbf{x}^k\right)^{\mathrm{T}}\mathbf{s}^k}{n}$, najdi řešení systému

$$\begin{pmatrix} \mathbf{0} & \mathbf{A}^{\mathrm{T}} & \mathbf{I} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{S}^k & \mathbf{0} & \mathbf{X}^k \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x}^k \\ \Delta\mathbf{y}^k \\ \Delta\mathbf{s}^k \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ -\mathbf{X}^k\mathbf{S}^k\mathbf{e} - \Delta\widetilde{\mathbf{X}}^k\Delta\widetilde{\mathbf{S}}^k\mathbf{e} + \iota_k\mathbf{e} \end{pmatrix}. \tag{3.18}$$

Vypočítej maximální délku kroku α_k splňující

$$\left(\mathbf{x}^k + \alpha_k\Delta\mathbf{x}^k, \mathbf{y}^k + \alpha_k\Delta\mathbf{y}^k, \mathbf{s}^k + \alpha_k\Delta\mathbf{s}^k\right) \in \mathscr{N}_{-\infty}(\gamma).$$

If $\alpha_k < \frac{27\gamma^2}{2n^2}$ **then**

pro $\iota_k = \frac{\beta}{1-\beta}\mu_k$, kde $\mu_k = \frac{\left(\mathbf{x}^k\right)^{\mathrm{T}}\mathbf{s}^k}{n}$, najdi řešení systému

$$\begin{pmatrix} \mathbf{0} & \mathbf{A}^{\mathrm{T}} & \mathbf{I} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{S}^k & \mathbf{0} & \mathbf{X}^k \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x}^k \\ \Delta\mathbf{y}^k \\ \Delta\mathbf{s}^k \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ -\mathbf{X}^k\mathbf{S}^k\mathbf{e} - \Delta\widetilde{\mathbf{X}}^k\Delta\widetilde{\mathbf{S}}^k\mathbf{e} + \iota_k\mathbf{e} \end{pmatrix}. \tag{3.19}$$

Vypočítej maximální délku kroku α_k splňující

$$\left(\mathbf{x}^k + \alpha_k\Delta\mathbf{x}^k, \mathbf{y}^k + \alpha_k\Delta\mathbf{y}^k, \mathbf{s}^k + \alpha_k\Delta\mathbf{s}^k\right) \in \mathscr{N}_{-\infty}(\gamma).$$

end (if)

Polož $\left(\mathbf{x}^{k+1}, \mathbf{y}^{k+1}, \mathbf{s}^{k+1}\right) = \left(\mathbf{x}^k + \alpha_k\Delta\mathbf{x}^k, \mathbf{y}^k + \alpha_k\Delta\mathbf{y}^k, \mathbf{s}^k + \alpha_k\Delta\mathbf{s}^k\right)$.

end (Korekční krok)

end (for)

end

Složitost Algoritmu MMA3 je opět polynomiální:

Věta 3.3. Algoritmus MMA3 končí nejvýše po $O\left(n^{\frac{5}{2}} log\frac{\left(\mathbf{x}^0\right)^{\mathrm{T}}\mathbf{s}^0}{\varepsilon}\right)$ iteracích s řešením, pro které platí $\mathbf{x}^{\mathrm{T}}\mathbf{s} \leq \varepsilon$.

Důkaz. viz [13]. □

3.2.4 Modifikace č. 4

Včleněním další regulace pro nalezení směru v závislosti na délce kroku vypočteného v „predikčním“ kroku do Algoritmu MMA3 je získán algoritmus vykazující v nejhorším případě nižší složitost:

Algoritmus MMA4

Vstupy:

parametr okolí $\gamma \in \left(0, \frac{1}{4}\right)$,

parametr záruky $\beta \in \left[\gamma, \frac{1}{4}\right)$,

startovací bod $\left(\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0\right) \in \mathscr{N}_{-\infty}(\gamma)$.

begin

for $k = 0, 1, 2, \ldots$

Predikční krok

Najdi řešení systému

$$\begin{pmatrix} \mathbf{0} & \mathbf{A}^{\mathrm{T}} & \mathbf{I} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{S}^k & \mathbf{0} & \mathbf{X}^k \end{pmatrix} \begin{pmatrix} \Delta\widetilde{\mathbf{x}}^k \\ \Delta\widetilde{\mathbf{y}}^k \\ \Delta\widetilde{\mathbf{s}}^k \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ -\mathbf{X}^k\mathbf{S}^k\mathbf{e} \end{pmatrix}.$$

Vypočítej maximální délku kroku $\widetilde{\alpha}_k$ splňující

$$\left(\mathbf{x}^k + \widetilde{\alpha}_k \Delta\mathbf{x}^k, \mathbf{y}^k + \widetilde{\alpha}_k \Delta\mathbf{y}^k, \mathbf{s}^k + \widetilde{\alpha}_k \Delta\mathbf{s}^k\right) \in \mathscr{F}.$$

end (Predikční krok)

Korekční krok

Spočítej $\widehat{\alpha}_k = 1 - \left(\frac{2\gamma t_k}{1-\gamma}\right)^{\frac{1}{3}}$, kde $t_k = \underset{i:\Delta\widetilde{x}_i^k \Delta\widetilde{s}_i^k > 0}{max} \left\{ \frac{\Delta\widetilde{x}_i^k \Delta\widetilde{s}_i^k}{x_i^k s_i^k} \right\}$.

If $\widetilde{\alpha}_k > \widehat{\alpha}_k$ **then**

položž $\widetilde{\alpha}_k = \widehat{\alpha}_k$.

end (if)

If $\widetilde{\alpha}_k \geq 0.1$ **then**

pro $\iota_k = (1-\widetilde{\alpha}_k)^3 \mu_k$, kde $\mu_k = \frac{\left(\mathbf{x}^k\right)^{\mathrm{T}} \mathbf{s}^k}{n}$, najdi řešení systému

$$\begin{pmatrix} \mathbf{0} & \mathbf{A}^{\mathrm{T}} & \mathbf{I} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{S}^k & \mathbf{0} & \mathbf{X}^k \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x}^k \\ \Delta\mathbf{y}^k \\ \Delta\mathbf{s}^k \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ -\mathbf{X}^k\mathbf{S}^k\mathbf{e} - \Delta\widetilde{\mathbf{X}}^k \Delta\widetilde{\mathbf{S}}^k\mathbf{e} + \iota_k\mathbf{e} \end{pmatrix}. \tag{3.20}$$

Vypočítej maximální délku kroku α_k splňující

$$\left(\mathbf{x}^k + \alpha_k \Delta\mathbf{x}^k, \mathbf{y}^k + \alpha_k \Delta\mathbf{y}^k, \mathbf{s}^k + \alpha_k \Delta\mathbf{s}^k\right) \in \mathscr{N}_{-\infty}(\gamma).$$

end (if)

If $\widetilde{\alpha}_k < 0.1$ **then**

pro $\iota_k = (1-\widetilde{\alpha}_k)^3 \mu_k$, kde $\mu_k = \frac{\left(\mathbf{x}^k\right)^{\mathrm{T}} \mathbf{s}^k}{n}$, najdi řešení systému

$$\begin{pmatrix} \mathbf{0} & \mathbf{A}^{\mathrm{T}} & \mathbf{I} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{S}^k & \mathbf{0} & \mathbf{X}^k \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x}^k \\ \Delta\mathbf{y}^k \\ \Delta\mathbf{s}^k \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ -\mathbf{X}^k\mathbf{S}^k\mathbf{e} - \widetilde{\alpha}_k \Delta\widetilde{\mathbf{X}}^k \Delta\widetilde{\mathbf{S}}^k\mathbf{e} + \iota_k\mathbf{e} \end{pmatrix}. \tag{3.21}$$

Vypočítej maximální délku kroku α_k splňující

$$\left(\mathbf{x}^k + \alpha_k \Delta\mathbf{x}^k, \mathbf{y}^k + \alpha_k \Delta\mathbf{y}^k, \mathbf{s}^k + \alpha_k \Delta\mathbf{s}^k\right) \in \mathscr{N}_{-\infty}(\gamma).$$

end (if)

If $\alpha_k < \frac{\gamma}{\sqrt{2n}}$ **then**

pro $\iota_k = \frac{\beta}{1-\beta}\mu_k$, kde $\mu_k = \frac{\left(\mathbf{x}^k\right)^{\mathrm{T}}\mathbf{s}^k}{n}$, najdi řešení systému

$$\begin{pmatrix} \mathbf{0} & \mathbf{A}^{\mathrm{T}} & \mathbf{I} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{S}^k & \mathbf{0} & \mathbf{X}^k \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x}^k \\ \Delta\mathbf{y}^k \\ \Delta\mathbf{s}^k \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ -\mathbf{X}^k\mathbf{S}^k\mathbf{e} - \widetilde{\alpha}_k\Delta\widetilde{\mathbf{X}}^k\Delta\widetilde{\mathbf{S}}^k\mathbf{e} + \iota_k\mathbf{e} \end{pmatrix}. \tag{3.22}$$

Vypočítej maximální délku kroku α_k splňující

$$\left(\mathbf{x}^k + \alpha_k\Delta\mathbf{x}^k, \mathbf{y}^k + \alpha_k\Delta\mathbf{y}^k, \mathbf{s}^k + \alpha_k\Delta\mathbf{s}^k\right) \in \mathscr{N}_{-\infty}(\gamma).$$

end (if)

Polož $\left(\mathbf{x}^{k+1}, \mathbf{y}^{k+1}, \mathbf{s}^{k+1}\right) = \left(\mathbf{x}^k + \alpha_k\Delta\mathbf{x}^k, \mathbf{y}^k + \alpha_k\Delta\mathbf{y}^k, \mathbf{s}^k + \alpha_k\Delta\mathbf{s}^k\right)$.

end (Korekční krok)

end (for)

end

Právě uvedený Algoritmus MMA4 oproti Algoritmu MMA3 vykazuje nižší teoretickou iterační složitost, jak dokládá následující věta:.

Věta 3.4. Algoritmus MMA4 končí nejvýše po $O\left(n^{\frac{3}{2}} log\frac{\left(\mathbf{x}^0\right)^{\mathrm{T}}\mathbf{s}^0}{\varepsilon}\right)$ iteracích s řešením, pro které platí $\mathbf{x}^{\mathrm{T}}\mathbf{s} \leq \varepsilon$.

Důkaz. viz [13].

□

Drobnými úpravami lze vytvořit další modifikace Mehrotrova algoritmu (viz např. [12]).

Vzhledem k obtížím s nalezením startovacího striktně přípustného bodu, bývají algoritmy modifikací Mehrotrova algoritmu upravovány tak, aby umožňovaly použití i nepřípustného startovacího bodu. Přechod od „přípustných“ variant metod k „nepřípustným“ je analogický tomu provedenému u metody sledování cesty s dlouhým krokem. Tato práce se „nepřípustnými“ variantami modifikací Mehrotrova algoritmu blíže nezabývá. Uvedené algoritmy využívá po menší úpravě, zapřičiňující neplatnost konvegenční teorie (viz následující kapitola), k otestování jejich chování na úlohách lineáního programování ze sady testovacích úloh NETLIB LP.

Kapitola 4

Realizace a numerické testování

Tato kapitola je věnována realizaci algoritmů z předcházejících kapitol a představení výsledků jejich aplikace na několik úloh lineárního programování (s řídkou maticí koeficientů).

4.1 Implementace algoritmů

4.1.1 Volba startovacího bodu

Počet iterací a tím i čas potřebný pro nalezení optimálního řešení úlohy závisí na volbě počátečního řešení $\left(\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0\right)$. V této práci je startovací bod určován následujícím výpočtem:

1. stanovení složek primární proměnné:

   Spočítej

   $$p_1 = max\left\{0.1, \frac{\max\limits_{i=1,\ldots,m} |b_i|}{m}\right\},$$

   $$\mathbf{A}\mathbf{x}^0 = \mathbf{b}.$$

   for $i = 1, ..., n$
   If $x_i^0 \leq p_1$ **then**
   polož $x_i^0 = p_1$.
   end (if)
   end (for)

2. stanovení složek duální proměnné:

   Spočítej

   $$p_2 = max\left\{0.1, \frac{\max\limits_{i=1,\ldots,n} |c_i|}{n}\right\},$$

   $$\mathbf{y}^0 = (0, ..., 0)^{\mathrm{T}},$$

   $$\mathbf{s}^0 = \mathbf{c} - \mathbf{A}^{\mathrm{T}}\mathbf{y}^0.$$

for $i = 1, ..., n$

If $s_i^0 \leq p_2$ **then**

položž $s_i^0 = p_2$.

end (if)

end (for)

Tato volba vyplývá ze snahy udržet hodnoty reziduí co nejmenší a současně „příliš“ neporušit centralitu.

4.1.2 Stanovení délky kroku

Délka kroku $\alpha_k \in (0,1]$ je v metodách určována tak, aby platilo $\mathbf{x}^{k+1} > \mathbf{0}$ a $\mathbf{s}^{k+1} > \mathbf{0}$, kde $\left(\mathbf{x}^{k+1}, \mathbf{y}^{k+1}, \mathbf{s}^{k+1}\right) = \left(\mathbf{x}^k + \alpha_k \Delta\mathbf{x}^k, \mathbf{y}^k + \alpha_k \Delta\mathbf{y}^k, \mathbf{s}^k + \alpha_k \Delta\mathbf{s}^k\right)$. Toho je dosaženo samostatným výpočtem kroků pro primární a duální proměné, přičemž pro jednu z jejich složek platí rovnost. Získané hodnoty jsou „odraženy“ od 0 vynásobením koeficientem τ blízkým 1. Požadavek, aby délka kroku nepřekročila hodnotu 1 (délka kroku v Newtovově metodě), zabezpečuje výběr minima z 1 a vypočtené hodnoty, jak ukazují následující vztahy

$$\alpha_k^{pri} = min\left(1, \tau \min_{i:\Delta x_i^k < 0} -\frac{x_i^k}{\Delta x_i^k}\right), \tag{4.1}$$

$$\alpha_k^{dual} = min\left(1, \tau \min_{i:\Delta s_i^k < 0} -\frac{s_i^k}{\Delta s_i^k}\right). \tag{4.2}$$

Jako výsledná délka kroku je volena ta menší z délek primárního a duálního kroku:

$$\alpha_k = min\left(\alpha_k^{pri}, \alpha_k^{dual}\right). \tag{4.3}$$

Základní Mehrotrův algoritmus pro tento postup představuje jistou výjimku, neboť nestanovuje α_k, ale ponechává délky kroků pro primární proměnnou $\mathbf{x}$ a duální proměnnou $\mathbf{s}$ zvlášť. Také při výpočtu délky kroku v „predikčním“ kroku není použit parametr τ (resp. jeho hodnota je rovna 1).

Případná podmínka setrvání každé iterace v okolí $\mathcal{N}_{-\infty}(\gamma)$ je realizována jednoduchým způsobem: vypočítaná délka α_k se zkracuje na polovinu, pokud existuje index i takový, že $\mathbf{x}_i^{k+1}\mathbf{s}_i^{k+1} < \gamma\mu_{k+1}$, přičemž protože pravá a levá strana nerovnosti závisí na hodnotě α_k , jsou v každém opakování cyklu přepočítávány, tedy:

while existuje $i : x_i^{k+1} s_i^{k+1} < \gamma \mu_{k+1}$ **do**

$$\begin{aligned} \alpha_k &= \tfrac{\alpha_k}{2}, \\ \mathbf{x}^{k+1} &= \mathbf{x}^k + \alpha_k \Delta \mathbf{x}^k, \\ \mathbf{s}^{k+1} &= \mathbf{s}^k + \alpha_k \Delta \mathbf{s}^k, \\ \mu_{k+1} &= \frac{\left(\mathbf{x}^{k+1}\right)^{\mathrm{T}} \mathbf{s}^{k+1}}{n}, \end{aligned}$$

end (while).

„Nepřípustné“ metody používají okolí $\mathcal{N}_{-\infty}(\gamma, \delta)$, ve kterém je navíc podmínka navíc $\|(\mathbf{r}_b, \mathbf{r}_c)\| \leq \frac{\|(\mathbf{r}_b^0, \mathbf{r}_c^0)\|}{\mu_0} \delta \mu$, kde parametr $\delta \geq 1$, zabraňující nárustu reziduí. Zajištění jejího splnění je realizováno zmenšováním α_k, dokud není nerovnost splněna, pomocí cyklu analogického předešlému:

while $\left\| \left(\mathbf{r}_b^{k+1}, \mathbf{r}_c^{k+1} \right) \right\| > \frac{\|(\mathbf{r}_b^0, \mathbf{r}_c^0)\|}{\mu_0} \delta \mu_{k+1}$ **do**

$$\begin{aligned} \alpha_k &= \tfrac{\alpha_k}{2}, \\ \mathbf{x}^{k+1} &= \mathbf{x}^k + \alpha_k \Delta \mathbf{x}^k, \\ \mathbf{y}^{k+1} &= \mathbf{y}^k + \alpha_k \Delta \mathbf{y}^k, \\ \mathbf{s}^{k+1} &= \mathbf{s}^k + \alpha_k \Delta \mathbf{s}^k, \\ \mathbf{r}_b^{k+1} &= \mathbf{A}\mathbf{x}^{k+1} - \mathbf{b}, \\ \mathbf{r}_c^{k+1} &= \mathbf{A}^{\mathrm{T}} \mathbf{y}^{k+1} + \mathbf{s}^{k+1} - \mathbf{c}, \\ \mu_{k+1} &= \frac{\left(\mathbf{x}^{k+1}\right)^{\mathrm{T}} \mathbf{s}^{k+1}}{n}, \end{aligned}$$

end (while).

Splnění Armiovy podmínky, popsané v Algoritmu NCDK, je realizováno obdobně:

while $\mu_{k+1} > (1 - 0{,}01\alpha_k)\, \mu_k$ **do**

$$\begin{aligned} \alpha_k &= \tfrac{\alpha_k}{2}, \\ \mathbf{x}^{k+1} &= \mathbf{x}^k + \alpha_k \Delta \mathbf{x}^k, \\ \mathbf{s}^{k+1} &= \mathbf{s}^k + \alpha_k \Delta \mathbf{s}^k, \\ \mu_{k+1} &= \frac{\left(\mathbf{x}^{k+1}\right)^{\mathrm{T}} \mathbf{s}^{k+1}}{n}, \end{aligned}$$

end (while).

4.1.3 Stanovení centrujícího parametru v metodě sledování cesty s dlouhým krokem

V metodě sledování cesty s dlouhým krokem je hodnota centrujícího parametru počítána adaptivně pomocí vzorce

$$\sigma_k = 0{,}1 \left(min \left\{ \left(0{,}05 \frac{1 - \xi_k}{\xi_k} \right)^3, 5 \right\} \right), \quad \text{kde } \xi = \frac{\min\limits_{i=1,\ldots,n} x_i^k s_i^k}{\mu_k}, \tag{4.4}$$

která zaručuje, že určená hodnota σ_k vždy leží v intervalu $(0, 0.5]$. Jedná se o drobnou modifikaci pravidla užívaného softwarem LOQO (viz [14]), jež zajišťuje, aby $\sigma_k \in (0, 0.8]$.

4.1.4 Stanovení směru

Nejnáročnější operací v algoritmech primárně-duálních metod vnitřních bodů je určení směru postupu k další iteraci, zejména proto, že matice koeficientů úloh lineárního programování bývá velkých rozměrů[1]. Situaci trochu zjednodušuje fakt, že je také řídká (stejně jako matice **A**). Určení směru v případě „nepřípustných" variant metod odpovídá nalezení řešení následující soustavy (označení iterace symbolem k je pro jednoduchost vynecháno):

$$\begin{pmatrix} \mathbf{0} & \mathbf{A}^{\mathrm{T}} & \mathbf{I} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{S} & \mathbf{0} & \mathbf{X} \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{y} \\ \Delta\mathbf{s} \end{pmatrix} = \begin{pmatrix} -\mathbf{r}_c \\ -\mathbf{r}_b \\ -\mathbf{r}_{xs} \end{pmatrix}, \tag{4.5}$$

kde $\mathbf{r}_{xs} = \mathbf{XSe} - \sigma\mu\mathbf{e}$ pro metodu sledování cesty s dlouhým krokem ,
$\mathbf{r}_{xs} = \mathbf{XSe}$ pro predikční krok Mehrotrova algoritmu a jeho modifikací,
$\mathbf{r}_{xs} = \mathbf{XSe} + \Delta\widetilde{\mathbf{X}}\Delta\widetilde{\mathbf{S}}\mathbf{e} - \sigma\mu\mathbf{e}$ pro korekční krok Mehrotrova algoritmu,
$\mathbf{r}_{xs} = \mathbf{XSe} + \Delta\widetilde{\mathbf{X}}\Delta\widetilde{\mathbf{S}}\mathbf{e} - \iota\mathbf{e}$ pro korekční krok modifikací Mehrotrova algoritmu č. 1, 3 a 4,
$\mathbf{r}_{xs} = \mathbf{XSe} + \widetilde{\alpha}_k\Delta\widetilde{\mathbf{X}}\Delta\widetilde{\mathbf{S}}\mathbf{e} - \iota\mathbf{e}$ pro korekční krok Mehrotrova algoritmu č. 2 a 4.
Uvedenou soustavu je možné řešit třemi ekvivalentními způsoby:

1. přímým výpočtem,

2. pomocí *rozšířeného systému* se symetrickou indefinitní maticí, který je vytvořen eliminací Δs ze soustavy (4.5). Tato úprava je možná, neboť **x** a **s** jsou kladné a diagonální matice **X** a **S** jsou regulární. Směr je potom získán jako řešení soustavy

$$\begin{aligned} \begin{pmatrix} \mathbf{0} & \mathbf{A} \\ \mathbf{A}^{\mathrm{T}} & -\mathbf{D}^{-2} \end{pmatrix} \begin{pmatrix} \Delta\mathbf{y} \\ \Delta\mathbf{x} \end{pmatrix} &= \begin{pmatrix} -\mathbf{r}_b \\ -\mathbf{r}_c + \mathbf{X}^{-1}\mathbf{r}_{xs} \end{pmatrix}, \\ \Delta\mathbf{s} &= -\mathbf{X}^{-1}\left(\mathbf{r}_{xs} + \mathbf{S}\Delta\mathbf{x}\right), \end{aligned} \tag{4.6}$$

   kde $\mathbf{D} = \mathbf{S}^{-\frac{1}{2}}\mathbf{X}^{\frac{1}{2}}$.

3. pomocí *systému normálních rovnic*, jež je vytvořen vyjádřením $\Delta\mathbf{x}$ ze soustavy(4.6), přičemž se využívá diagonality a regularity matice $\mathbf{X}^{-1}\mathbf{S}$. Určení směru pak spočívá

[1] matice **A** totiž sama bývá velkých rozměrů

ve vyřešení soustavy

$$\begin{aligned} \mathbf{A}\mathbf{D}^2\mathbf{A}^{\mathrm{T}}\Delta\mathbf{y} &= -\mathbf{r}_b + \mathbf{A}\left(-\mathbf{S}^{-1}\mathbf{X}\mathbf{r}_c + \mathbf{S}^{-1}\mathbf{r}_{xs}\right), \\ \Delta\mathbf{s} &= -\mathbf{r}_c - \mathbf{A}^{\mathrm{T}}\Delta\mathbf{y}, \\ \Delta\mathbf{x} &= -\mathbf{S}^{-1}\left(\mathbf{r}_{xs} + \mathbf{X}\Delta\mathbf{s}\right), \end{aligned} \tag{4.7}$$

kde matice koeficientů $\mathbf{A}\mathbf{D}^2\mathbf{A}^{\mathrm{T}}$ je pozitivně semidefinitní.

Všechny tři způsoby řešení soustavy (4.5) jsou spojeny s problémy stability (zejména v závěrečných fázích výpočtu algoritmů je indikována špatná podmíněnost nebo singularita) v důsledku přítomnosti velmi malých a velmi velkých diagonálních prvků v maticích $\mathbf{D}$ a $\mathbf{D}^{-1}$, rovněž také kvůli hodnosti matice $\mathbf{A}$, která nemusí být plná. Udává se (např. v [2]), že formulace (4.7) pro nalezení směru se stává neúčinná zejména, pokud matice $\mathbf{A}\mathbf{S}^{-1}\mathbf{X}\mathbf{A}^{\mathrm{T}}$ je mnohem hustší než matice $\mathbf{A}$. To nastane v případě, že $\mathbf{A}$ obsahuje jeden nebo více hustých sloupců.

4.1.5 Poznámky k modifikacím Mehrotrova algoritmu

Výše uvedená schémata modifikací Mehrotrova algoritmu uvažují striktně přípustný startovací bod $\left(\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0\right) \in \mathcal{N}_{-\infty}(\gamma)$, a proto při hledání dalších iterací využívají soustavy pro určení směru, ve kterých jsou ve vektoru pravých stran obsaženy nulové vektory. V programových kódech vytvořených v programu MATLAB 7.5.0 (R2007b) je však použit nepřípustný startovací bod $\left(\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0\right)$, kde $\mathbf{x}^0 > 0$ a $\mathbf{s}^0 > 0$. Následkem toho jsou v soustavách pro určení směru upraveny pravé strany, a sice nahrazením nulových vektorů vektory $-\mathbf{r}_c^k$ a $-\mathbf{r}_b^k$. Tato samotná úprava bez dalších změn způsobí, že se na algoritmy již nevztahuje konvergenční teorie.

4.1.6 Ukončovací kriteria

Algoritmy uvedené v předešlých kapitolách předpokládají existenci optimálního primárně-duálního řešení (nejsou v nich tedy žádné mechanismy umožňující jejich ukončení dříve než je nalezeno). V realizaci je jako prvek ukončující činnost algoritmů bez nalezení řešení včleněno testování podmínky $\left\|\left(\mathbf{x}^k, \mathbf{s}^k\right)\right\|_\infty \geq K \left\|\left(\mathbf{x}^0, \mathbf{s}^0\right)\right\|_\infty$, kde $K = 10^{15}$. Po jejím splnění dojde k zastavení výpočtu s výpisem

"!!! ŘEŠENÍ NENALEZENO !!!"

Nenastane-li takováto situace, je jako ukončovací kriterium použito splnění podmínky $\left(\mathbf{x}^k\right)^{\mathrm{T}}\mathbf{s}^k \leq \varepsilon$, kde $\varepsilon = 10^{-6}$.

V některých případech může být iterační výpočet velmi zdouhavý, proto je v realizacích algoritmů jednotlivých metod stanoven maximální počet iterací, po jehož dosažení je výpočet ukončen, aniž je splněna kterákoli z výše uvedených podmínek.

4.2 Programové kódy metod

Algoritmy metod vnitřních bodů uvedených výše jsou zpracovány v programu MATLAB 7.5.0 (R2007b) do podoby funkcí uváděných v tzv. m-souborech.

4.2.1 Popis funkcí jednotlivých metod

Funkce Metoda_C_cesty_DK_A_r.m

Popis: „nepřípustná“ varianta metody sledování cesty s dlouhým krokem, přičemž směr postupu k další iteraci je určován přímým výpočtem.
Volené parametry: gama, K, maxiter, vypis, adaptivni, příp. tau, delta.
Zkratka: NCDK

Funkce Metoda_C_cesty_DK_RS_A_r.m

Popis: „nepřípustná“ varianta metody sledování cesty s dlouhým krokem, přičemž směr postupu k další iteraci je určován pomocí rozšířeného systému.
Volené parametry: gama, K, maxiter, vypis, adaptivni, příp. tau, delta.
Zkratka: NCDK_RS

Funkce Metoda_C_cesty_DK_NR_A_r.m

Popis: „nepřípustná“ varianta metody sledování cesty s dlouhým krokem, přičemž směr postupu k další iteraci je určován pomocí systému normálních rovnic.
Volené parametry: gama, K, maxiter, vypis, adaptivni, příp. tau, delta.
Zkratka: NCDK_NR

Funkce Funkce Metoda_C_cesty_DK.m

Popis: „nepřípustná“ varianta základního primárně-duálního algoritmu („nepřípustná“ varianta metody sledování cesty s dlouhým krokem bez aplikace okolí a Armiovy podmínky), přičemž směr postupu k další iteraci je určován přímým výpočtem.
Volené parametry: K, maxiter, vypis, adaptivni, příp. tau.
Zkratka: PD

Funkce Metoda_C_cesty_DK_RS.m

Popis: „Nepřípustná“ varianta základního primárně-duálního algoritmu („nepřípustná“ varianta metody sledování cesty s dlouhým krokem bez aplikace okolí a Armiovy podmínky), přičemž směr postupu k další iteraci je určován pomocí rozšířeného systému.
Volené parametry: K, maxiter, vypis, adaptivni, příp. tau.
Zkratka: PD_RS

Funkce Metoda_C_cesty_DK_NR.m

Popis: „Nepřípustná“ varianta základního primárně-duálního algoritmu („nepřípustná“ varianta metody sledování cesty s dlouhým krokem bez aplikace okolí a Armiovy podmínky), přičemž směr postupu k další iteraci je určován pomocí systému normálních rovnic.
Volené parametry: K, maxiter, vypis, adaptivni, příp. tau.
Zkratka: PD_NR

Funkce Mehrotrova_metoda.m

Popis: Mehrotrův algoritmus, přičemž směr postupu k další iteraci je určován přímým výpočtem .
Volené parametry: K, maxiter, vypis, adaptivni, příp. tau.
Zkratka: MA

Funkce Mehrotrova_metoda_RS.m

Popis: Mehrotrův algoritmus, přičemž směr postupu k další iteraci je určován pomocí rozšířeného systému.
Volené parametry: K, maxiter, vypis, adaptivni, příp. tau.
Zkratka: MA_RS

Funkce Mehrotrova_metoda_NR.m

Popis: Mehrotrův algoritmus, přičemž směr postupu k další iteraci je určován pomocí systému normálních rovnic.
Volené parametry: K, maxiter, vypis, adaptivni, příp. tau.
Zkratka: MA_NR

Funkce Mehrotrova_metoda_Modif_A1_2005.m

Popis: modifikace Mehrotrova algoritmu č. 1, přičemž směr postupu k další iteraci je určován přímým výpočtem.

Volené parametry: gamma, K, maxiter, vypis, adaptivni, příp. tau.
Zkratka: MMA1

Funkce Mehrotrova_metoda_Modif_A1_2005_RS.m

Popis: modifikace Mehrotrova algoritmu č. 1, přičemž směr postupu k další iteraci je určován pomocí rozšířeného systému.
Volené parametry: gamma, K, maxiter, vypis, adaptivni, příp. tau.
Zkratka: MMA1_RS

Funkce Mehrotrova_metoda_Modif_A1_2005_NR.m

Popis: modifikace Mehrotrova algoritmu č. 1, přičemž směr postupu k další iteraci je určován pomocí systému normálních rovnic.
Volené parametry: gamma, K, maxiter, vypis, adaptivni, příp. tau.
Zkratka: MMA1_NR

Funkce Mehrotrova_metoda_Modif_A1M_2005.m

Popis: modifikace Mehrotrova algoritmu č. 2, přičemž směr postupu k další iteraci je určován přímým výpočtem.
Volené parametry: gamma, K, maxiter, vypis, adaptivni, příp. tau.
Zkratka: MMA2

Funkce Mehrotrova_metoda_Modif_A1M_2005_RS.m

Popis: modifikace Mehrotrova algoritmu č. 2, přičemž směr postupu k další iteraci je určován pomocí rozšířeného systému.
Volené parametry: gamma, K, maxiter, vypis, adaptivni, příp. tau.
Zkratka: MMA2_RS

Funkce Mehrotrova_metoda_Modif_A1M_2005_NR.m

Popis: modifikace Mehrotrova algoritmu č. 2, přičemž směr postupu k další iteraci je určován pomocí systému normálních rovnic.
Volené parametry: gamma, K, maxiter, vypis, adaptivni, příp. tau.
Zkratka: MMA2_NR

Funkce Mehrotrova_metoda_Modif_A1_2008.m

Popis: modifikace Mehrotrova algoritmu č. 3, přičemž směr postupu k další iteraci je určován přímým výpočtem.

Volené parametry: gamma, beta, K, maxiter, vypis, adaptivni, příp. tau.
Zkratka: MMA3

Funkce Mehrotrova_metoda_Modif_A1_2008_RS.m

Popis: modifikace Mehrotrova algoritmu č. 3, přičemž směr postupu k další iteraci je určován pomocí rozšířeného systému.
Volené parametry: gamma, beta, K, maxiter, vypis, adaptivni, příp. tau.
Zkratka: MMA3_RS

Funkce Mehrotrova_metoda_Modif_A1_2008_NR.m

Popis: modifikace Mehrotrova algoritmu č. 3, přičemž směr postupu k další iteraci je určován pomocí systému normálních rovnic.
Volené parametry: gamma, beta, K, maxiter, vypis, adaptivni, příp. tau.
Zkratka: MMA3_NR

Funkce Mehrotrova_metoda_Modif_A2_2008.m

Popis: modifikace Mehrotrova algoritmu č. 4, přičemž směr postupu k další iteraci je určován přímým výpočtem.
Volené parametry: gamma, beta, K, maxiter, vypis, adaptivni, příp. tau.
Zkratka: MMA4

Funkce Mehrotrova_metoda_Modif_A2_2008_RS.m

Popis: modifikace Mehrotrova algoritmu č. 4, přičemž směr postupu k další iteraci je určován pomocí rozšířeného systému.
Volené parametry: gamma, beta, K, maxiter, vypis, adaptivni, příp. tau.
Zkratka: MMA4_RS

Funkce Mehrotrova_metoda_Modif_A2_2008_NR.m

Popis: modifikace Mehrotrova algoritmu č. 4, přičemž směr postupu k další iteraci je určován pomocí systému normálních rovnic.
Volené parametry: gamma, beta, K, maxiter, vypis, adaptivni, příp. tau.
Zkratka: MMA4_NR

4.2.2 Popis volání funkcí

Funkce jednotlivých metod jsou volány pomocí příkazu

```
x=název_funkce(A,b,c,options),
```

kde název_funkce je název jedné z výše uvedených funkcí,

A je matice úlohy lineárního programování,

b, c jsou vektory úlohy lineárního programování,

options je struktura volitelných parametrů metody.

4.2.3 Nastavování volitelných parametrů

Nastavení hodnot volitelných parametrů se provádí následovně

```
options=impset('vlastnost',hodnota,...)
```

kde impset(·) je funkce vytvořená úpravou funkce ODESET(·) z programu MATLAB 7.5.0 (R2007b) (viz help programu MATLAB),

options je struktura tvořená složkami popsanými v následující tabulce:

Tabulka 4.1: Složky struktury options

Složka	Popis	Povolené hodnoty	Implicitní nastavení
gama	parametr okolí užívaný v NCDK, NCDK_RS, NCDK_NR	$(0,1)$	10^{-3}
gamma	parametr okolí užívaný v MMA1, MMA1_RS, MMA1_NR, MMA2, MMA2_RS, MMA2_NR, MMA3, MMA3_RS, MMA3_NR, MMA4, MMA4_RS, MMA4_NR	$\left(0,\frac{1}{4}\right)$	10^{-3}
beta	parametr záruky užívaný v MMA3, MMA3_RS, MMA3_NR, MMA4, MMA4_RS, MMA4_NR	$[\text{gamma},\frac{1}{4})$	$\frac{1}{5}$
epsilon	přesnost výpočtu	malé kladné reálné číslo	10^{-6}
K	parametr pro vyhodnocení neexistence řešení úlohy	kladné celé číslo	10^{15}
maxiter	povolený maximální počet iterací	kladné celé číslo	1000
vypis	parametr pro zákaz/povolení výpisu vypočítaných údajů v každé iteraci	$\{0,1\}$	0
adaptivni	parametr pro zapnutí/vypnutí adaptivní volby tau a delta	$\{0,1\}$	0
tau	parametr ovlivňující délku kroku	$[0.9,1)$	0.9
delta	parametr okolí pro „nepřípustné“ varianty metod užívaný v NCDK, NCDK_RS, NCDK_NR	$[1,\infty)$	10

4.3 Numerické výsledky

4.3.1 Testovací úlohy

Testování programových kódů jednotlivých metod uvedených výše, vytvořených v programu MATLAB 7.5.0 (R2007b), probíhá na vybraných úlohách sady testovacích problémů NETLIB LP (viz http://www.netlib.org/lp/index.html) uvedených v tabulce 4.2.

Tabulka 4.2: Testovací úlohy LP_NETLIB

Název úlohy	Vlastnosti matice A			Deklarovaná hodnota účelové funkce
	Počet řádků	Počet sloupců	Počet nenulových prvků	
lp_adlittle	56	138	138	2.2549496316E+05
lp_afiro	27	51	102	-4.6475314286E+02
lp_agg	488	615	2 862	-3.5991767287E+07
lp_d2q06c	2 171	5 831	3 3081	1.2278423615E+05
lp_ship04l	402	2 166	6 380	1.7933245380E+06
lp_ship04s	402	1 506	4 400	1.7987147004E+06
lp_ship08l	778	4 363	12 882	1.9090552114E+06
lp_ship08s	778	2 467	7 194	1.9200982105E+06
lp_ship12l	1 151	5 533	16 276	1.4701879193E+06
lp_ship12s	1 151	2 869	8 284	1.4892361344E+06
lpi_bgprtr	20	40	70	————*
lpi_itest6	11	17	29	————*
Poznámka: * úloha nemá optimální řešení.				

4.3.2 Volba parametrů pro testování

Rychlost výpočtu, příp. selhání algoritmů je odvislé mj. od hodnot parametrů, proto je třeba věnovat jejich volbě dostatečnou pozornost. V této práci je postupováno následovně: Za hodnotu parametru okolí γ je zvolena obvyklá hodnota, tj. $\gamma = 10^{-3}$. Pro ni jsou otestovány různé kombinace hodnot ostatních parametrů. A na základě chování algoritmů metod je stanovena tzv. základní volba parametrů, jedná se o hodnoty, při nichž algoritmy fungují pro většinu testovaných úloh. Poté jsou postupně měněny hodnoty jednotlivých parametrů a studovány projevy, těchto změn. Následně jsou představeny hodnoty parametrů (v rámci jejich testovaných hodnot), pro něž algoritmy dosahují nejlepších výsledků, zde označované jako nejlepší volba parametrů. Protože se při řešení úloh objevují jisté problémy, jsou také nabízena možná řešení, která vyústí v adaptivní volbu některých parametrů.

4.3.3 Výsledky pro základní volbu parametrů

Vhodnou základní volbou parametrů je $\gamma = 10^{-3}$, $\delta = 10$, $\tau = 0.9$, $\beta = \frac{1}{5}$, $K = 10^{15}$ a $\varepsilon = 10^{-6}$.

Testovací úloha lp_adlittle

Metoda sledování cesty s dlouhým krokem a základní primárně-duální algoritmus vykazují při řešení úlohy lp_adlittle nejvyšší počet iterací potřebný pro nalezení ε-přesného řešení, ale protože se v každé iteraci řeší pouze jediná soustava pro určení směru, jsou tyto metody z časového hlediska srovnatelné s Mehrotrovým algoritmem a jeho modifikacemi, v nichž jsou v jednotlivých iteracích řešeny soustavy dvě. Použití systému normálních rovnic pro určení směru se jeví v případě všech metod jako velmi výhodné, neboť se tím snižuje doba výpočtu.

Tabulka 4.3: Výsledky pro testovací úlohy lp_adlittle a lp_afiro při základní volbě parametrů

	lp_adlittle		lp_afiro	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	30	0.166011	17	0.061247
NCDK_RS	30	0.096201	17	0.052555
NCDK_NR	30	0.078750	17	0.051301
PD	30	0.153832	17	0.057937
PD_RS	30	0.089242	17	0.050548
PD_NR	30	0.081807	17	0.047165
MA	22	0.214893	12	0.066814
MA_RS	22	0.120832	12	0.054588
MA_NR	22	0.097553	12	0.051005
MMA1	24	0.213796	13	0.071023
MMA1_RS	24	0.129846	13	0.054636
MMA1_NR	24	0.109833	13	0.055955
MMA2	23	0.207489	14	0.074599
MMA2_RS	23	0.125363	14	0.059889
MMA2_NR	23	0.107167	14	0.057435
MMA3	24	0.216548	13	0.078076
MMA3_RS	24	0.129440	13	0.058801
MMA3_NR	24	0.111119	13	0.055537
MMA4	23	0.210204	13	0.072370
MMA4_RS	23	0.125846	13	0.057747
MMA4_NR	23	0.133754	13	0.056326

Testovací úloha lp_afiro

Pro úlohu lp_afiro s maticí koeficientů $\mathbf{A}$ poměrně malých rozměrů, platí v podstatě totéž, co pro úlohu lp_adlittle.

Testovací úloha lp_agg

Při řešení úlohy lp_agg uvedenými metodami vnitřních bodů nastávají výpočetní problémy v případě použití systému normálních rovnic pro určení směru v metodě sledování cesty s dlouhým krokem (NCDK_NR) a základním primárně-duálním algoritmu (PD_NR). Během výpočtu PD_NR program Matlab 7.5.0 (R2007b) vypíše, že hodnota účelové funkce i míry duality je NaN (Not-a-Number)[2]. Ani v ostatních metodách se aplikace systému normálních rovnic pro určení směru nejeví jako výhodné, protože se tím výrazně zvyšuje počet iterací potřebných pro nalezení ε-přesného řešení, a tím se prodlužuje i doba výpočtu. Nejlepších výsledků z hlediska počtu iterací i času je dosaženo Mehrotrovým algoritmem.

Tabulka 4.4: Výsledky pro testovací úlohy lp_agg a lp_d2q06c při základní volbě parametrů

	lp_agg		lp_d2q06c	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	150	5.324811	88	155.222846
NCDK_RS	150	2.880378	88	18.762213
NCDK_NR	10 000*	408.421373	161	69.490212
PD	148	4.814120	84	146.800629
PD_RS	148	2.647984	84	18.420236
PD_NR	10 000*	21 195.616921	92	25.766696
MA	53	2.666887	48	186.729835
MA_RS	53	1.421567	48	20.730962
MA_NR	58	2.682151	51	28.011797
MMA1	66	3.332977	110	344.789915
MMA1_RS	64	1.746444	96	37.490286
MMA1_NR	154	8.837334	114	54.791494
MMA2	203	9.234192	653	1 874.927178
MMA2_RS	203	5.515957	653	302.377782
MMA2_NR	244	10.608097	658	345.960032
MMA3	60	2.884311	59	209.236012
MMA3_RS	60	1.567546	59	25.056311
MMA3_NR	146	10.699405	62	34.394256
MMA4	59	2.871880	53	194.181809
MMA4_RS	59	1.524144	53	23.206499
MMA4_NR	95	5.757624	56	32.571565
Poznámka: * iterační výpočet ukončen před nalezením ε-přesného řešení				

Testovací úloha lp_d2q06c

Na základě výsledků získaných aplikací jednotlivých metod na úlohu lp_d2q06c je možné učinit závěr, že pro určení směru je u všech metod nejvýhodnější použít rozšířený systém, neboť ε-přesné řešení je pak nalezeno rychleji než při užití přímého výpočtu (při stejném, nebo

[2]Pro vysvětlení viz help programu Matlab.

nižším počtu iterací), a také rychleji než při použití systému normálních rovnic (při nižším počtu iterací). Nejkratší dobu výpočtu vykazuje metoda sledování cesty s dlouhým krokem a základní primárně-duální algoritmus používající právě rozšířený systém (tj. NCDK_RS a PD_RS), byť počet iterací při jejich aplikaci není nejnižší. Doba výpočtu varianty Mehrotrova algoritmu stanovující směr pomocí rozšířeného systému (MA_RS), získaná s nižším počtem iterací, je také poměrně krátká.

Testovací úloha lp_ship04l

Při řešení úlohy lp_ship04l jednotlivými metodami se ukazuje jako výhodné užít pro určení směru systém normálních rovnic. Tato strategie sice počet iterací potřebných pro nalezení ε-přesného řešení, v porovnání s ostatními způsoby stanovení směru, nesníží (výjimkou je srovnání MMA1_NR a MMA1_RS), ale zkracuje dobu výpočtu. Pořadí metod, stanovujících směr pomocí systému normálních, dle doby výpočtu od nejkratší po nejdelší vypadá následovně:

MMA4_NR, PD_NR, MA_NR, MMA3_NR, NCDK_NR, MMA1_NR a MMA2_NR.

Tabulka 4.5: Výsledky pro testovací úlohy lp_ship04l a lp_ship04s při základní volbě parametrů

	lp_ship04l		lp_ship04s	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	58	8.004368	71	5.317426
NCDK_RS	58	2.071377	71	1.703349
NCDK_NR	58	1.036704	75	0.860625
PD	55	7.512500	69	5.106917
PD_RS	55	1.929808	69	1.617936
PD_NR	55	0.938186	69	0.786650
MA	29	7.943988	33	4.912245
MA_RS	29	1.992358	33	1.586894
MA_NR	29	0.958045	33	0.743717
MMA1	36	9.884194	40	5.969333
MMA1_RS	37	2.472949	40	1.915789
MMA1_NR	36	1.092687	40	0.896270
MMA2	61	16.406255	66	9.795041
MMA2_RS	61	4.101146	66	3.125191
MMA2_NR	61	1.772260	66	1.436199
MMA3	32	8.649426	35	5.183459
MMA3_RS	32	2.145920	35	1.675427
MMA3_NR	32	0.986550	35	0.798457
MMA4	30	8.128638	36	5.307760
MMA4_RS	30	2.034057	36	1.725420
MMA4_NR	30	0.923815	36	0.814783

Testovací úloha lp_ship04s

Počet iterací potřebných k nalezení ε-přesného řešení úlohy lp_ship04s je při použití přímého výpočtu soustavy pro určení směru, rozšířeného systému a systému normálních rovnic v rámci jednotlivých metod stejný s jedinou výjimkou, a sice metody sledování cesty s dlouhým krokem. Z časového hlediska je výhodné použít k nalezení směru systém normálních rovnic. Toto platí pro všechny metody. Při aplikaci kterékoli metody, je-li směr hledán pomocí systému normálních rovnic, je výpočet poměrně rychlý, přesto nejkratší dobu výpočtu udává MA_NR, která vykazuje také nejnižší počet iterací. Použití PD_NR se z hlediska času jeví také velmi výhodné.

Testovací úloha lp_ship08l

Při řešení úlohy lp_ship08l pomocí uvedených metod je situace obdobná jako v případě úlohy lp_ship04s. Výjimku zde tvoří Modifikace Mehrotrova algoritmu č. 1. Nejvýhodnější je z hlediska doby výpočtu použít pro nalezení ε-přesného řešení MA_NR, která vykazuje také nejnižší počet iterací, přip. modifikaci MMA3_NR.

Tabulka 4.6: Výsledky pro testovací úlohy lp_ship08l a lp_ship08s při základní volbě parametrů

	lp_ship08l		lp_ship08s	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	77	21.371969	73	7.859535
NCDK_RS	77	5.344283	73	2.841248
NCDK_NR	77	2.380944	73	1.348446
PD	69	19.025598	69	7.396232
PD_RS	69	4.715919	69	2.625497
PD_NR	69	2.093808	69	1.227193
MA	31	17.142313	33	7.070627
MA_RS	31	4.295941	33	2.538275
MA_NR	31	1.917851	33	1.185616
MMA1	39	21.576432	46	9.835511
MMA1_RS	41	5.644719	46	3.553331
MMA1_NR	39	2.349510	46	1.621802
MMA2	72	39.709467	59	12.580133
MMA2_RS	72	9.778588	59	4.547968
MMA2_NR	72	4.165235	59	2.059433
MMA3	32	17.704319	40	8.578224
MMA3_RS	32	4.442437	40	3.090423
MMA3_NR	32	1.974394	40	1.433657
MMA4	39	21.560303	35	7.504646
MMA4_RS	39	5.350821	35	2.704611
MMA4_NR	39	2.362211	35	1.278289

Testovací úloha lp_ship08s

Na základě výsledků získaných aplikací jednotlivých metod na úlohu lp_ship08s je možné říci, že z hlediska počtu iterací je pro jednotlivé metody lhostejné, zda je pro nalezení směru užit přímý výpočet soustavy pro určení směru, rozšířený systém, nebo systém normálních rovnic. Z časového hlediska je nejvýhodnější použít systém normálních rovnic. Ze všech uvedených metod je pro výpočet nejvýhodnější zvolit MA_NR, příp. PD_NR.

Testovací úloha lp_ship12l

V tomto případě je situace obdobná jako u úlohy lp_ship04s. Výjimku zde tvoří modifikace Mehrotrova algoritmu č. 2. Opět je nejvýhodnější použít k nalezení ε-přesného řešení MA_NR, příp. PD_NR.

Tabulka 4.7: Výsledky pro testovací úlohy lp_ship12l a lp_ship12s při základní volbě parametrů

	lp_ship12l		lp_ship12s	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	74	22.560795	64	6.762548
NCDK_RS	74	6.497408	65	3.045373
NCDK_NR	74	2.904813	64	1.434702
PD	65	19.701658	56	5.882228
PD_RS	65	5.618268	56	2.560983
PD_NR	65	2.481739	56	1.217489
MA	32	19.517358	32	6.715752
MA_RS	32	5.594810	32	2.926204
MA_NR	32	2.456912	32	1.368552
MMA1	50	30.538922	52	10.886814
MMA1_RS	50	8.646467	53	4.837535
MMA1_NR	50	3.742413	52	2.195952
MMA2	64	38.878320	140	28.963169
MMA2_RS	64	10.984614	140	12.362824
MMA2_NR	65	4.817151	140	5.785728
MMA3	38	23.233908	41	8.589559
MMA3_RS	38	6.649872	41	3.797444
MMA3_NR	38	2.902072	41	1.764593
MMA4	41	24.877533	39	8.170335
MMA4_RS	41	7.129456	39	3.614470
MMA4_NR	41	3.127272	39	1.699396

Testovací úloha lp_ship12s

Stejně jako pro většinu již uvedených úloh je i pro lp_ship12s výhodné při hledání ε-přesného řešení aplikovat metodu, v níž je pro určení směru využíván systém normálních

rovnic. Přitom z časového hlediska se jako nejvhonější jeví použití PD_NR, ačkoliv počet iterací potřebný k nalezení optimálního řešení není nejnižší. Nejnižší počet iterací vykazuje Mehrotrův algoritmus, ale pro nalezení ε-přesného řešení potřebuje více času. Je to způsobeno tím, že Mehrotrův algoritmus probíhá ve dvou krocích, přičemž v každém se řeší soustava rovnic, narozdíl od základního primárně-duálního algoritmu, kde je v každé iteraci počítáno řešení jen jediné soustavy.

Testovací úloha lpi_bgprtr

Všechny uvedené metody až na metodu sledování cesty, která určuje směr postupu k další iteraci pomocí systému normálních rovnic (tj. NCDK_NR), identifikují neexistenci optimálního řešení úlohy lpi_bgprtr v menším počtu iterací než 10 000. Přitom v nejnižším počtu iterací toto rozpozná varianta Mehrotrova algoritmu, v níž je směr stanoven řešením rozšířeného systému (tj. MA_RS), v nejkratším čase také varianta Mehrotrova algoritmu ale ta, v níž je směr určován řešením systému normálních rovnic (tj. MA_NR).

Tabulka 4.8: Výsledky pro testovací úlohy lpi_bgprtr a lpi_itest6 při základní volbě parametrů

	lpi_bgprtr		lpi_itest6	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	123	0.266643	134	0.177257
NCDK_RS	102	0.132192	134	0.134760
NCDK_NR	10 000*	30.315618	10 000*	8.857725
PD	66	0.140550	80	0.105526
PD_RS	66	0.088314	80	0.083303
PD_NR	61	0.082092	6 603	4.481774
MA	15	0.074059	33	0.094358
MA_RS	12	0.055791	33	0.071124
MA_NR	14	0.052461	33	0.071181
MMA1	25	0.110446	30	0.083318
MMA1_RS	22	0.072800	30	0.068588
MMA1_NR	28	0.074325	30	0.069398
MMA2	73	0.278929	75	0.168241
MMA2_RS	60	0.135174	75	0.127202
MMA2_NR	144	0.266795	255	0.333703
MMA3	22	0.101764	35	0.101071
MMA3_RS	23	0.077883	35	0.079914
MMA3_NR	30	0.081369	35	0.074602
MMA4	93	0.502091	73	0.236150
MMA4_RS	57	0.171257	73	0.167024
MMA4_NR	86	0.214531	227	0.424922
Poznámka: * iterační výpočet ukončen předčasně bez rozpoznání neexistence řešení				

Testovací úloha lpi_itest6

Stejně jako při řešení úlohy lpi_bgprtr, se použití NCDK_NR pro rozpoznání neexistence optimálního řešení úlohy lpi_itest6 nejeví jako vhodné, protože se jí to nepodaří ani v 10 000 iteracích. Naopak vhodnou metodou je MA_RS, která odhalí neexistenci optimálního řešení s nejnižším počtem iterací a současně i v nejkratším čase.

Zhodnocení metod při užití základní volby parametrů

Z uvedených výsledků plyne, že doba výpočtu je do značné míry ovlivněna způsobem hledání směru. Zatímco stanovování směru pomocí přímého výpočtu soustavy pro určení směru, nebo pomocí rozšířeného systému vedlo bezpečně k správné identifikaci (ne)existence optimálního řešené úlohy, příp. k nalezení ε-přesného řešení, použití rozšířeného systému vykazovalo v této oblasti jisté těžkosti. Lze tedy doporučit využívání rozšířeného systému, který znamená kratší dobu výpočtu než v případě použití přímého výpočtu. Z metod je vhodné využívat Mehrotův algoritmus, který zpravidla vykazuje nejnižší počet iterací, ovšem ne vždy nejvyšší čas, a funguje spolehlivě i s normálním systémem rovnic. Konvergence této metody je v porovnání s jeho modifikacemi rychlejší.

4.3.4 Výsledků metod při změnách hodnoty parametru γ

Nyní bude předmětem zájmu chování metod při změně hodnoty parametru γ, jednak při zvýšení z hodnoty 10^{-3} na 10^{-2} a jednak při snížení z 10^{-3} na 10^{-4}. Hodnoty ostatních parametrů přitom zůstanou nezměněny, tj. stejné jako v případě základní volby parametrů $(\delta = 10, \tau = 0.9, \beta = \frac{1}{5}, K = 10^{15}$ a $\varepsilon = 10^{-6})$.

Testovací úloha lp_adlittle

Zvýšení hodnoty parametru γ z obvyklé hodnoty 10^{-3} na 10^{-2} vede u metody sledování cesty s dlouhým krokem (NCDK, NCDK_RS a NCDK_NR) ke zvýšení počtu iterací a tím i k prodloužení času potřebného k nalezení ε-přesného řešení úlohy lp_adlittle. Naopak snížení hodnoty parametru γ z obvyklé hodnoty na 10^{-4} nemá vliv na změnu počtu iterací, ale vede k zkrácení doby potřebné pro výpočet. V porovnání s ostatními metodami je počet iterací u této metody vyšší, ale doba výpočtu je v některých případech nižší. Při aplikaci ostatních uvedených metod se změna hodnoty parametru γ na počtu iterací neprojeví, ale na době výpočtu ano. Pro MMA1_RS, MMA2, MMA3 a MMA4 je nejvýhodnější použít $\gamma = 10^{-2}$, pro MMA1, MMA1_NR, MMA2_RS, MMA2_NR, MMA3_NR a MMA4_RS $\gamma = 10^{-3}$ a pro MMA3_RS a MMA4_NR $\gamma = 10^{-4}$.

Tabulka 4.9: Výsledky pro testovací úlohu lp_adlittle při různých hodnotách parametru γ

	$\gamma = 10^{-2}$		$\gamma = 10^{-3}$		$\gamma = 10^{-4}$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	31	0.205492	30	0.166011	30	0.165303
NCDK_RS	31	0.101567	30	0.096201	30	0.095974
NCDK_NR	31	0.083030	30	0.078750	30	0.078462
MMA1	24	0.256217	24	0.213796	24	0.219840
MMA1_RS	24	0.128324	24	0.129846	24	0.128909
MMA1_NR	24	0.110836	24	0.109833	24	0.110837
MMA2	23	0.206911	23	0.207489	23	0.208381
MMA2_RS	23	0.127414	23	0.125363	23	0.125635
MMA2_NR	23	0.109851	23	0.107167	23	0.107704
MMA3	24	0.214017	24	0.216548	24	0.214237
MMA3_RS	24	0.129741	24	0.129440	24	0.128793
MMA3_NR	24	0.111801	24	0.111119	24	0.111304
MMA4	23	0.207997	23	0.210204	23	0.208871
MMA4_RS	23	0.126845	23	0.125846	23	0.125927
MMA4_NR	23	0.133893	23	0.133754	23	0.132243

Testovací úloha lp_afiro

Počet iterací potřebný k nalezení ε-přesného řešení úlohy lp_afiro se při změně hodnoty parametru γ nemění. Z hlediska doby výpočtu je pro většinu uvedených metod nejvýhodnější zvolit $\gamma = 10^{-4}$. Při obvyklé volbě γ je nejkratší doba výpočtu jen pro NCDK_NR a MMA2.

Tabulka 4.10: Výsledky pro testovací úlohu lp_afiro při různých hodnotách parametru γ

	$\gamma = 10^{-2}$		$\gamma = 10^{-3}$		$\gamma = 10^{-4}$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	17	0.057010	17	0.061247	17	0.060881
NCDK_RS	17	0.055940	17	0.052555	17	0.052087
NCDK_NR	17	0.131234	17	0.051301	17	0.052074
MMA1	13	0.106751	13	0.071023	13	0.069154
MMA1_RS	13	0.054464	13	0.054636	13	0.054890
MMA1_NR	13	0.136460	13	0.055955	13	0.054189
MMA2	14	0.077779	14	0.074599	14	0.074834
MMA2_RS	14	0.060020	14	0.059889	14	0.059771
MMA2_NR	14	0.057537	14	0.057435	14	0.055163
MMA3	13	0.072913	13	0.078076	13	0.077621
MMA3_RS	13	0.062856	13	0.058801	13	0.058152
MMA3_NR	13	0.055304	13	0.055537	13	0.054816
MMA4	13	0.072251	13	0.072370	13	0.071903
MMA4_RS	13	0.058093	13	0.057747	13	0.057547
MMA4_NR	13	0.055246	13	0.056326	13	0.055022

Testovací úloha lp_agg

Na základě výsledků z níže uvedené tabulky pro úlohu lp_agg je možné učinit tento závěr: při snižování hodnoty parametru γ, počet iterací klesá, nebo zůstává stejný, přičemž výjimkami jsou MMA1_NR, MMA2, MMA2_RS, MMA3_NR a MMA4_NR a samozřejmě NCDK_NR, jež se pro řešení úlohy nejeví jako vhodná (jsou ukončeny předšasně bez nalezení řešení z důvodu „nepřijatelně dlouhé“ doby trvání iteračního výpočtu, což ani jiná volba hodnoty γ nemění). Zatímco pro MMA4_NR je nejvhodnější volba $\gamma = 10^{-3}$, a to jak z hlediska počtu iterací, tak i doby výpočtu, pro MMA1_NR, MMA2, MMA2_RS a MMA3_NR tomu tak není. Pro MMA1_NR a MMA3_NR je z předkládaných voleb nejlepší $\gamma = 10^{-4}$, pro MMA2 a MMA2_RS $\gamma = 10^{-2}$.

Tabulka 4.11: Výsledky pro testovací úlohu lp_agg při různých hodnotách parametru γ

	$\gamma = 10^{-2}$		$\gamma = 10^{-3}$		$\gamma = 10^{-4}$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	178	6.368602	150	5.324811	148	5.348890
NCDK_RS	178	3.359769	150	2.880378	148	2.857506
NCDK_NR	10 000*	419.299065	10 000*	408.421373	10 000*	431.234504
MMA1	69	3.544392	66	3.332977	66	3.273672
MMA1_RS	68	1.907240	64	1.746444	62	1.760745
MMA1_NR	119	6.425645	154	8.837334	106	5.594146
MMA2	153	7.052637	203	9.234192	185	8.353905
MMA2_RS	153	4.032939	203	5.515957	185	5.042009
MMA2_NR	630	37.277387	244	10.608097	204	8.505143
MMA3	61	3.017431	60	2.884311	60	2.938563
MMA3_RS	61	1.609907	60	1.567546	60	1.584771
MMA3_NR	115	7.751197	146	10.699405	92	5.615467
MMA4	64	3.097374	59	2.871880	59	2.913413
MMA4_RS	64	1.663186	59	1.524144	59	1.522850
MMA4_NR	114	7.404397	95	5.757624	100	6.425427
Poznámka: * iterační výpočet ukončen před nalezením ε-přesného řešení						

Testovací úloha lp_d2q06c

Při aplikaci modifikací Mehrotrova algoritmu č. 1 a 2 (to je MMA1, MMA1_RS, MMA1_NR, MMA2, MMA2_RS a MMA2_NR) na úlohu lp_d2q06c je vhodnější volit nižší hodnotu γ, zatímco pro modifikace č. 3 a 4 vyšší (MMA3, MMA3_RS, MMA3_NR, MMA4, MMA4_RS a MMA4_NR). Zajímavé je, že použití různých postupů určení směru u metody sledování cesty s dlouhým krokem (viz NCDK, NCDK_RS a NCDK_NR), vede při změně hodnoty parametru γ k různým tendencím ve změně počtu iterací. V případě NCDK a NCDK_RS při zvýšení hodnoty γ počet iterací klesá, ale v případě NCDK_NR se nejprve zvýší a potom klesne.

Tabulka 4.12: Výsledky pro testovací úlohu lp_d2q06c při různých hodnotách parametru γ

	$\gamma = 10^{-2}$		$\gamma = 10^{-3}$		$\gamma = 10^{-4}$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	93	160.259518	88	155.222846	84	146.411314
NCDK_RS	93	19.791715	88	18.762213	84	18.107377
NCDK_NR	108	31.686947	161	69.490212	117	42.449904
MMA1	92	294.192246	110	344.789915	109	344.596800
MMA1_RS	89	35.211158	96	37.490286	109	42.229174
MMA1_NR	98	50.153192	114	54.791494	112	53.004283
MMA2	248	764.478079	653	1 874.927178	1 230	3 554.081316
MMA2_RS	248	109.021017	653	302.377782	1 220	597.194541
MMA2_NR	255	129.972550	658	345.960032	1 221	682.566788
MMA3	61	218.140611	59	209.236012	55	198.820246
MMA3_RS	61	25.689546	59	25.056311	55	23.397161
MMA3_NR	67	41.518508	62	34.394256	59	35.219917
MMA4	56	203.707148	53	194.181809	50	189.581331
MMA4_RS	56	24.209113	53	23.206499	50	22.003258
MMA4_NR	61	38.145380	56	32.571565	53	31.661040

Testovací úloha lp_ship04l

Při určování ε-přesného řešení úlohy lp_ship04l pomocí metody sledování cesty s dlouhým krokem (NCDK, NCDK_RS a NCDK_NR) a modifikace Mehrotrova algoritmu č. 1 (MMA1, MMA1_RS a MMA1_NR) platí, že čím je hodnota parametru γ menší, tím je počet iterací nižší, a tím i doba potřebná pro nalezení ε-přesného řešení kratší. Modifikace Mehrotrova algoritmu č. 2 (MMA2, MMA2_RS a MMA2_NR) vykazuje pro obvyklou volbu parametru γ nejvyšší počet iterací, a tím i nejdelší dobu dobu výpočtu. Nejlépe se tato metoda chová, je-li $\gamma = 10^{-4}$. Modifikace Mehrotrova algoritmu č. 4 (MMA4, MMA4_RS a MMA4_NR) naopak při volbě parametru $\gamma = 10^{-3}$ vyžaduje k nalezení ε-přesného řešení nejnižší počet iterací i nejkratší dobu výpočtu. Modifikace Mehrotrova algoritmu č. 3 (MMA3, MMA3_RS a MMA3_NR) vykazuje pro $\gamma = 10^{-3}$ a $\gamma = 10^{-4}$ stejný počet iterací, přičemž doba výpočtu se liší jen mírně.

Tabulka 4.13: Výsledky pro testovací úlohu lp_ship04l při různých hodnotách parametru γ

	$\gamma = 10^{-2}$		$\gamma = 10^{-3}$		$\gamma = 10^{-4}$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	65	9.174480	58	8.004368	55	7.532204
NCDK_RS	65	2.368280	58	2.071377	55	1.948801
NCDK_NR	65	1.107635	58	1.036704	55	0.963009
MMA1	37	10.032258	36	9.884194	34	9.223730
MMA1_RS	39	2.598511	37	2.472949	35	2.333405
MMA1_NR	37	1.114812	36	1.092687	34	1.034081
MMA2	55	14.810021	61	16.406255	51	13.732051
MMA2_RS	55	3.730744	61	4.101146	51	3.431183
MMA2_NR	55	1.611605	61	1.772260	51	1.495419
MMA3	33	8.955095	32	8.649426	32	8.652343
MMA3_RS	33	2.221159	32	2.145920	32	2.144202
MMA3_NR	33	1.010182	32	0.986550	32	0.989124
MMA4	33	8.925715	30	8.128638	31	8.385312
MMA4_RS	33	2.221802	30	2.034057	31	2.097999
MMA4_NR	33	1.004183	30	0.923815	31	0.955825

Testovací úloha lp_ship04s

Pro NCDK, NCDK_RS, NCDK_NR, MMA1, MMA1_NR, MMA2, MMA2_RS a MMA2_NR při hledání ε-přesného řešení úlohy lp_ship04s platí, že čím je hodnota parametru γ menší, tím je počet iterací nižší, a tím i doba výpočtu kratší. Modifikace Mehrotrova algoritmu č. 2 (MMA2, MMA2_RS a MMA2_NR) vykazuje pro obvyklou volbu parametru $\gamma = 10^{-3}$ nejvyšší počet iterací, a tím i nejdelší dobu výpočtu. Naopak nejlépe se tato metoda chová, je-li $\gamma = 10^{-4}$. Jako v předešlé úloze potřebuje modifikace Mehrotrova algoritmu č. 3 (MMA3, MMA3_RS a MMA3_NR) k nalezení ε-přesného řešení pro $\gamma = 10^{-3}$ a $\gamma = 10^{-4}$ stejný počet iterací, přičemž doba výpočtu se mírně liší. Zde navíc platí totéž pro MMA1_RS. Modifikace Mehrotrova algoritmu č. 4 (MMA4, MMA4_RS a MMA4_NR) vykazuje při obvyklé volbě parametru γ nejnižší počet iterací i nejkratší dobu výpočtu.

Tabulka 4.14: Výsledky pro testovací přiklad lp_ship04s pro různé hodnoty parametru γ

	$\gamma = 10^{-2}$		$\gamma = 10^{-3}$		$\gamma = 10^{-4}$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	84	6.339069	71	5.317426	69	5.156197
NCDK_RS	84	1.974537	71	1.703349	69	1.663665
NCDK_NR	86	0.996941	75	0.860625	72	0.835564
MMA1	41	6.101057	40	5.969333	37	5.517629
MMA1_RS	43	2.070587	40	1.915789	40	1.934128
MMA1_NR	41	0.917395	40	0.896270	37	0.851524
MMA2	68	10.089858	66	9.795041	62	9.198825
MMA2_RS	68	3.252639	66	3.125191	62	2.954670
MMA2_NR	68	1.461475	66	1.436199	62	1.340311
MMA3	36	5.345702	35	5.183459	35	5.176251
MMA3_RS	36	1.725506	35	1.675427	35	1.666030
MMA3_NR	36	0.808365	35	0.798457	35	0.790617
MMA4	38	5.590305	36	5.307760	37	5.465562
MMA4_RS	38	1.804674	36	1.725420	37	1.762650
MMA4_NR	38	0.863972	36	0.814783	37	0.841364

Testovací úloha lp_ship08l

Metoda sledování cesty s dlouhým krokem (NCDK, NCDK_RS a NCDK_NR) opět při snížení hodnoty parametru γ z obvyklé hodnoty na 10^{-4} potřebuje k nalezení ε-přesného řešení úlohy lp_ship08l nižší počet iterací i kratší dobu, při zvýšení hodnoty parametru na 10^{-2} zvýšení počtu iterací a prodloužení doby. Obvyklá volba parametru $\gamma = 10^{-3}$ se jeví jako nejvýhodnější v případě použití modifikace Mehrotrova algoritmu č. 3 (MMA3, MMA3_RS a MMA3_NR) a modifikace Mehrotrova algoritmu č. 1, je-li směr určován pomocí rozšířeného systému (MMA1_RS). Vhodná je také pro modifikaci Mehrotrova algoritmu č. 4 (MMA4, MMA4_RS a MMA4_NR), která pro $\gamma = 10^{-3}$ a $\gamma = 10^{-4}$ nalézá ε-přesné řešení se stejným počtem iterací. Jako nejméně vhodná je volba $\gamma = 10^{-3}$ pro modifikaci Mehrotrova algoritmu č. 2 (MMA2, MMA2_RS a MMA2_NR).

Tabulka 4.15: Výsledky pro testovací úlohu lp_ship08l při různých hodnotách parametru γ

	$\gamma = 10^{-2}$		$\gamma = 10^{-3}$		$\gamma = 10^{-4}$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	83	23.049697	77	21.371969	74	20.508674
NCDK_RS	83	5.744268	77	5.344283	74	5.141981
NCDK_NR	83	2.535137	77	2.380944	74	2.297377
MMA1	41	22.662748	39	21.576432	39	21.570311
MMA1_RS	43	5.903641	41	5.644719	42	5.751236
MMA1_NR	41	2.467019	39	2.349510	39	2.345138
MMA2	68	37.500764	72	39.709467	67	36.903744
MMA2_RS	68	9.265727	72	9.778588	67	9.155194
MMA2_NR	68	3.952047	72	4.165235	67	3.915837
MMA3	36	19.984561	32	17.704319	35	19.337093
MMA3_RS	36	5.007923	32	4.442437	35	4.844795
MMA3_NR	36	2.195956	32	1.974394	35	2.140918
MMA4	40	22.127922	39	21.560303	39	21.497125
MMA4_RS	40	5.476866	39	5.350821	39	5.341907
MMA4_NR	40	2.415837	39	2.362211	39	2.362725

Testovací úloha lp_ship08s

Většina metod pro nižší hodnoty parametru γ vykazuje nižší počet iterací pro nalezení ε-přesného řešení úlohy lp_ship08s, a tím i kratší dobu výpočtu. Pro modifikaci Mehrotrova algoritmu č. 2 (MMA2, MMA2_RS a MMA2_NR) je tomu opačně, tedy snižení hodnoty parametru γ vyžaduje vyšší počet iterací i delší dobu výpočtu. Zvláštním případem je modifikace Mehrotrova algoritmu č. 4 (MMA4, MMA4_RS a MMA4_NR), která pro $\gamma = 10^{-3}$ a $\gamma = 10^{-4}$ určí ε-přesného řešení se stejným počtem iterací.

Tabulka 4.16: Výsledky pro testovací úlohu lp_ship08s při různých hodnotách parametru γ

	$\gamma = 10^{-2}$		$\gamma = 10^{-3}$		$\gamma = 10^{-4}$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	75	8.070843	73	7.859535	71	7.644184
NCDK_RS	75	2.932535	73	2.841248	72	2.793137
NCDK_NR	75	1.378386	73	1.348446	71	1.313260
MMA1	47	10.043705	46	9.835511	44	9.428963
MMA1_RS	47	3.634590	46	3.553331	44	3.376325
MMA1_NR	47	1.661845	46	1.621802	44	1.560050
MMA2	57	12.164209	59	12.580133	60	12.800689
MMA2_RS	57	4.383107	59	4.547968	60	4.568066
MMA2_NR	57	2.000596	59	2.059433	60	2.101425
MMA3	41	8.768618	40	8.578224	38	8.152938
MMA3_RS	41	3.178933	40	3.090423	38	2.940589
MMA3_NR	41	1.468104	40	1.433657	38	1.378649
MMA4	36	7.733906	35	7.504646	35	7.520646
MMA4_RS	36	2.774494	35	2.704611	35	2.693977
MMA4_NR	36	1.306974	35	1.278289	35	1.276549

Testovací úloha lp_ship12l

Při snížení hodnoty parametru γ z hodnoty 10^{-3} na 10^{-4} metoda sledování cesty s dlouhým krokem (NCDK, NCDK_RS a NCDK_NR) a modifikace Mehrotrova algoritmu. č. 4 (MMA4, MMA4_RS a MMA4_NR) vykazují nižší počet iterací potřebný k nalezení ε-přesného řešení úlohy lp_ship12l i kratší dobu výpočtu, při zvýšení γ z hodnoty 10^{-3} na 10^{-2} vyšší počet iterací a delší dobu výpočtu. Změna hodnoty γ nemá vliv na počet iterací v_případě použití MMA1 nebo MMA1_NR. Pro modifikaci Mehrotrova algoritmu č. 2 (MMA2, MMA2_RS a MMA2_NR) se jako nejvhonější volba jeví $\gamma = 10^{-2}$, neboť pro tuto hodnotu je počet iterací nejnižší a doba výpočtu nejkratší. Při aplikaci MMA1_RS a MMA3, MMA3_RS nebo MMA3_NR se při snížení hodnoty parametru γ z obvyklé volby na hodnotu 10^{-4} počet iterací potřebný k nalezení ε-přesného řešení řešení nezmění (doba výpočtu se mění jen nepartně) a přitom je nižší než pro $\gamma = 10^{-2}$.

Tabulka 4.17: Výsledky pro testovací příklad lp_ship12l pro různé hodnoty parametru γ

	$\gamma = 10^{-2}$		$\gamma = 10^{-3}$		$\gamma = 10^{-4}$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	87	26.516167	74	22.560795	70	21.328968
NCDK_RS	87	7.630147	74	6.497408	70	6.138041
NCDK_NR	87	3.368568	74	2.904813	70	2.743928
MMA1	50	30.527428	50	30.538922	50	30.532187
MMA1_RS	51	8.825095	50	8.646467	50	8.681391
MMA1_NR	50	3.723503	50	3.742413	50	3.714473
MMA2	59	35.796460	64	38.878320	66	40.076254
MMA2_RS	59	10.156778	64	10.984614	66	11.279265
MMA2_NR	59	4.401088	65	4.817151	66	4.881701
MMA3	42	25.588702	38	23.233908	38	23.166372
MMA3_RS	42	7.340157	38	6.649872	38	6.605525
MMA3_NR	42	3.178127	38	2.902072	38	2.916403
MMA4	43	26.123908	41	24.877533	40	24.275351
MMA4_RS	43	7.492596	41	7.129456	40	6.956253
MMA4_NR	43	3.279818	41	3.127272	40	3.056270

Testovací úloha lp_ship12s

Při snížení hodnoty parametru γ z hodnoty 10^{-3} na 10^{-4} se při aplikaci metody sledování cesty s dlouhým krokem (NCDK, NCDK_RS a NCDK_NR) a modifikace Mehrotrova algoritmu č. 4 (MMA4, MMA4_RS a MMA4_NR) sníží počet iterací potřebný k určení ε-přesného řešení úlohy lp_ship12s a zkrátí se i doba výpočtu. Při zvýšení z 10^{-3} na 10^{-2} se počet iterací zvýší a doba výpočtu se prodlouží. Obvyklá hodnota γ, tj. 10^{-3}, vykazuje v MMA1 nebo MMA1_NR nevyšší počet iterací, zatímco pro $\gamma = 10^{-2}$ a $\gamma = 10^{-4}$ je počet iterací stejný. Z hlediska doby trvání je ale vhodnější použít vyšší z hodnot. Při aplikaci modifikace Mehrotrova algoritmu č. 2 (MMA2, MMA2_RS a MMA2_NR) se jako nejvýhodnější volba jeví, stejně jako u předcházející úlohy, $\gamma = 10^{-2}$, neboť pro tuto hodnotu je počet iterací nejnižší a doba výpočtu nejkratší. V případě použití modifikace Mehrotrova algoritmu č. 3 (MMA3, MMA3_RS a MMA3_NR) je nejvhodnější použít $\gamma = 10^{-4}$.

Tabulka 4.18: Výsledky pro testovací přiklad lp_ship12s pro různé hodnoty parametru γ

	$\gamma = 10^{-2}$		$\gamma = 10^{-3}$		$\gamma = 10^{-4}$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	72	7.622072	64	6.762548	61	6.459129
NCDK_RS	73	3.403581	65	3.045373	62	2.902618
NCDK_NR	72	1.599110	64	1.434702	61	1.366963
MMA1	51	10.682329	52	10.886814	51	10.684806
MMA1_RS	51	4.662496	53	4.837535	50	4.578675
MMA1_NR	51	2.147841	52	2.195952	51	2.151182
MMA2	60	12.506499	140	28.963169	142	29.388360
MMA2_RS	60	5.483086	140	12.362824	142	12.446651
MMA2_NR	60	2.542746	140	5.785728	142	5.868342
MMA3	41	8.610575	41	8.589559	39	8.182198
MMA3_RS	41	3.789200	41	3.797444	39	3.584463
MMA3_NR	41	1.752435	41	1.764593	39	1.667322
MMA4	41	8.604706	39	8.170335	38	7.985873
MMA4_RS	41	3.761728	39	3.614470	38	3.462298
MMA4_NR	41	1.767979	39	1.699396	38	1.645604

Testovací úloha lpi_bgprtr

Změna hodnoty parametru γ nemá v případě NCDK_NR za následek rozpoznání neexistence řešení úlohy lpi_bgprtr do 10 000. iterace. Pro MMA1, MMA2_NR, MMA4 a MMA4_NR je pro rozpoznání neexistence řešení nejvýhodnější, z hlediska počtu iterací i doby výpočtu, použít za hodnotu parametru γ 10^{-2}, pro NCDK 10^{-3} a pro NCDK_RS, MMA1_RS, MMA2_RS, MMA3 a MMA4_RS 10^{-4}. MMA1_NR udává pro $\gamma = 10^{-2}$ a $\gamma = 10^{-3}$ stejné počty iterací, a přitom vyšší než pro hodnotu $\gamma = 10^{-4}$. Při aplikaci MMA2 jsou počty iterací pro $\gamma = 10^{-2}$ a $\gamma = 10^{-4}$ stejné, ale z hlediska doby výpočtu se jeví jako vhodnější volba první z uvedených. Modifikace Mehrotrova algoritmu č. 3 (MMA3, MMA3_RS a MMA3_NR) vykazuje pro $\gamma = 10^{-3}$ a $\gamma = 10^{-4}$ stejné počty iterací, ale z hlediska doby výpočtu je pro MMA3 a MMA3_RS výhodnější zvolit $\gamma = 10^{-4}$ a pro $\gamma = 10^{-3}$.

Tabulka 4.19: Výsledky pro testovací úlohu lpi_bgprtr při různých hodnotách parametru γ

	$\gamma = 10^{-2}$		$\gamma = 10^{-3}$		$\gamma = 10^{-4}$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	125	0.384224	123	0.266643	124	0.272138
NCDK_RS	104	0.133449	102	0.132192	102	0.130789
NCDK_NR	10 000*	27.168741	10 000*	30.315618	10 000*	29.898415
MMA1	24	0.101547	25	0.110446	36	0.147713
MMA1_RS	25	0.073203	22	0.072800	17	0.060244
MMA1_NR	28	0.076444	28	0.074325	27	0.075553
MMA2	70	0.262450	73	0.278929	70	0.271482
MMA2_RS	61	0.136333	60	0.135174	59	0.133402
MMA2_NR	96	0.187693	144	0.266795	101	0.196008
MMA3	26	0.114546	22	0.101764	22	0.101127
MMA3_RS	26	0.075558	23	0.077883	23	0.075536
MMA3_NR	26	0.076771	30	0.081369	30	0.080947
MMA4	75	0.399489	93	0.502091	82	0.449113
MMA4_RS	58	0.175147	57	0.171257	56	0.170899
MMA4_NR	45	0.128628	86	0.214531	50	0.134960
Poznámka: * iterační výpočet ukončen předčasně bez rozpoznání neexistence řešení						

Testovací úloha lpi_itest6

Pro většinu metod je nejvhodnější použít $\gamma = 10^{-4}$, která přináší identifikaci neexistence řešení úlohy lpi_itest6 v nejmenším počtu iterací, a tím i v nejkratším čase. Při aplikaci modifikace Mehrotrova algorimu č. 1 je lepší zvolit $\gamma = 10^{-3}$, zatímco při užití modifikace Mehrotrova algoritmu č. 3 $\gamma = 10^{-2}$. Ani při změně hodnoty parametru γ není v případě NCDK_NR rozpoznána neexistence řešení do 10 000. iterace.

Tabulka 4.20: Výsledky pro testovací úlohy lpi_itest6 při různých hodnotách parametru γ

	$\gamma = 10^{-2}$		$\gamma = 10^{-3}$		$\gamma = 10^{-4}$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	138	0.207607	134	0.177257	132	0.170553
NCDK_RS	138	0.136347	134	0.134760	132	0.132200
NCDK_NR	10 000*	8.818288	10 000*	8.857725	10 000*	8.426129
MMA1	32	0.088503	30	0.083318	27	0.081269
MMA1_RS	32	0.076386	30	0.068588	27	0.066209
MMA1_NR	32	0.072193	30	0.069398	31	0.071404
MMA2	76	0.168562	75	0.168241	74	0.169537
MMA2_RS	76	0.127852	75	0.127202	74	0.123820
MMA2_NR	225	0.298613	255	0.333703	177	0.238664
MMA3	26	0.085473	35	0.101071	32	0.090984
MMA3_RS	26	0.068757	35	0.079914	32	0.075714
MMA3_NR	26	0.066271	35	0.074602	30	0.079365
MMA4	74	0.228696	73	0.236150	71	0.219904
MMA4_RS	74	0.169861	73	0.167024	71	0.162726
MMA4_NR	279	0.532707	227	0.424922	218	0.378996
Poznámka: * iterační výpočet ukončen předčasně bez rozpoznání neexistence řešení						

Zhodnocení metod při změnách hodnoty parametru γ

Nelze dopředu říci, která z předkládaných hodnot parametru γ bude pro řešení úlohy nejvhodnější. Pouze metoda sledování cesty s dlouhým krokem (není-li pro nalezení směru použit systém normálních rovnic) vykazuje pro všechny testovací úlohy a testované hodnoty γ jistou stabilní tendenci spočívající v tom, že při snížení hodnoty parametru γ, klesne počet iterací (potřebný pro nalezení ε-přesného řešení v případě, že ho úloha má optimální řešení, nebo odhalení neexistence optimálního řešení, pokud ho úloha nemá), nebo se nezmění. Doba vyžadovaná pro výpočet se tedy se snížením hodnoty γ, zkrátí, sníží-li se počet iterací. Nezmění-li se počet iterací, změní se doba pouze nepatrně a je tedy zpravidla lhostejné, která z hodnot parametru γ, bude pro výpočet použita.

Dále je z testování hodnot γ patrné, že je-li rychlost konveregence „nízká“, změna γ v uvedeném rozsahu situaci výrazně nezmění.

4.3.5 Výsledky metod při změně hodnoty parametru τ

Základní volbou parametru τ byla stanovena hodnota 0.9, což je jeho dolní mez. Nyní bude testováno chování jednotlivých metod při jejím zvýšení na hodnotu blízkou jeho horní mezi, a sice $\tau = 0.999$. Hodnoty ostatních parametrů přitom opět zůstanou nezměněny, tj. stejné jako v případě základní volby parametrů ($\delta = 10$, $\gamma = 10^{-3}$, $\beta = \frac{1}{5}$, $K = 10^{15}$ a $\varepsilon = 10^{-6}$) .

Testovací úloha lp_adlittle

Zvýšení hodnoty parametru τ z 0.9 na 0.999 vede při řešení úlohy lp_adlittle pomocí Mehrotrova algoritmu (MA, MA_RS a MA_NR) a jeho modifikací (MMA1, MMA1_RS, MMA1_NR, MMA2, MMA2_RS, MMA2_NR, MMA3, MMA3_RS, MMA3_NR, MMA4, MMA4_RS a MMA4_NR) k snížení počtu iterací i zkrácení doby výpočtu. Všechny modifikace dokonce vykazují stejný počet iterací. Jejich doby výpočtu, jsou-li pro $\tau = 0.999$ srovnávány verze určující směr stejným postupem (tj. přímým výpočtem soustavy pro stanovení směru, pomocí rozšířeného systému, nebo pomocí systému normálních rovnic) se liší jen nepatrně. Volba $\tau = 0.999$ se nejeví jako vhodná pro metodu sledování cesty s dlouhým krokem (NCDK, NCDK_RS a NCDK_NR) a pro základní primárně-duální algoritmus (PD, PD_RS a PD_NR). U první jmenované dojde k zhoršení výpočetních vlastností v tom smyslu, že ani v 10 000. iteracích nenalezne ε-přesné řešení úlohy. Toto je možné, při zachování $\tau = 0.999$, napravit volbou $\gamma = 10^{-2}$. V případě aplikace základního primárně-duálního algoritmu dochází k špatné identifikaci (ne)existence optimálního řešení úlohy, což volba $\gamma = 10^{-2}$ nebo $\gamma = 10^{-4}$, ani zvýšení hodnoty parametru K na 10^{35} nezmění.

Tabulka 4.21: Výsledky pro testovací úlohy lp_adlittle při různých hodnotách parametru τ

	$\tau = 0.9$		$\tau = 0.999$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	30	0.166011	10 000*	112.325578
NCDK_RS	30	0.096201	10 000*	58.801359
NCDK_NR	30	0.078750	10 000*	54.399322
PD	30	0.153832	17**	0.154589
PD_RS	30	0.089242	17**	0.127904
PD_NR	30	0.081807	17**	0.115345
MA	22	0.214893	15	0.156794
MA_RS	22	0.120832	15	0.092538
MA_NR	22	0.097553	15	0.077692
MMA1	24	0.213796	19	0.171680
MMA1_RS	24	0.129846	19	0.108019
MMA1_NR	24	0.109833	19	0.094728
MMA2	23	0.207489	19	0.180498
MMA2_RS	23	0.125363	19	0.109070
MMA2_NR	23	0.107167	19	0.094962
MMA3	24	0.216548	19	0.180089
MMA3_RS	24	0.129440	19	0.109536
MMA3_NR	24	0.111119	19	0.096351
MMA4	23	0.210204	19	0.175505
MMA4_RS	23	0.125846	19	0.109311
MMA4_NR	23	0.133754	19	0.119807

Poznámky: * iterační výpočet ukončen před nalezením ε-přesného řešení
** výpočet ukončen s hlášením: „!!! ŘEŠENÍ NELZE NAJÍT !!!“

Testovací úloha lp_afiro

Jen v případě aplikace metody sledování cesty s dlouhým krokem se při zvýšení hodnoty parametru τ z 0.9 na 0.999 počet iterací potřebných k nalezení k nalezení ε-přesného řešení úlohy lp_afiro zvyšuje. U ostatních metod je tedy výhodnější použít $\tau = 0.999$, a to jak z hlediska počtu iterací, tak z hlediska doby výpočtu.

Tabulka 4.22: Výsledky pro testovací úlohu lp_afiro při různých hodnotách parametru τ

	$\tau = 0.9$		$\tau = 0.999$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	17	0.061247	19	0.065911
NCDK_RS	17	0.052555	19	0.052517
NCDK_NR	17	0.051301	19	0.049941
PD	17	0.057937	13	0.051309
PD_RS	17	0.050548	13	0.043436
PD_NR	17	0.047165	13	0.042634
MA	12	0.066814	7	0.054598
MA_RS	12	0.054588	7	0.045982
MA_NR	12	0.051005	7	0.044463
MMA1	13	0.071023	9	0.057604
MMA1_RS	13	0.054636	9	0.047770
MMA1_NR	13	0.055955	9	0.048791
MMA2	14	0.074599	10	0.064415
MMA2_RS	14	0.059889	10	0.052873
MMA2_NR	14	0.057435	10	0.052371
MMA3	13	0.078076	9	0.062845
MMA3_RS	13	0.058801	9	0.050760
MMA3_NR	13	0.055537	9	0.048580
MMA4	13	0.072370	9	0.060498
MMA4_RS	13	0.057747	9	0.050528
MMA4_NR	13	0.056326	9	0.048811

Testovací úloha lp_agg

Problémy s nalezením ε-přesného řešení úlohy lp_agg, se objevují při aplikaci metody sledování cesty s dlouhým krokem (NCDK, NCDK_RS a NCDK_NR) a základního primárně-duálního algoritmu (PD, PD_RS a PD_NR). Použít NCDK_NR a PD_NR na úlohu lp_agg není vhodné, jak pro $\tau = 0.9$, tak $\tau = 0.999$. Přitom během výpočtu PD_NR program Matlab 7.5.0 (R2007b) vypíše, že hodnota účelové funkce a míry duality je NaN (tj. Not-a-Number)[3]. Iterační výpočet NCDK, NCDK_RS a NCDK_NR pro $\tau = 0.999$ byl ukončen před nalezením ε-přesného řešení z důvodu vysoké náročnosti výpočtu na čas. Řešením tohoto problému může být volba $\gamma = 10^{-2}$. Stejně jako pro úlohu lp_adlittle dochází při aplikaci PD a PD_RS k špatné identifikaci (ne)existence optimálního řešení, nápravou zde může být zvýšení hodnoty parametru K na 10^{35} .

Pro Mehrotrovu metodu a její modifikace je volba $\tau = 0.999$ výhodná, neboť při ní vykazují nižší počet iterací a kratší dobu výpočtu než pro $\tau = 0.9$, výjimky tvoří pouze MMA1 a MMA4_NR.

[3]Pro vysvětlení viz help programu Matlab.

Tabulka 4.23: Výsledky pro testovací úlohy lp_agg při různých hodnotách parametru τ

	$\tau = 0.9$		$\tau = 0.999$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	150	5.324811	2 000*	5 512.354690
NCDK_RS	150	2.880378	2 000*	633.353024
NCDK_NR	10 000*	408.421373	2 000*	1 359.667897
PD	148	4.814120	146**	4.558318
PD_RS	148	2.647984	143**	1.843072
PD_NR	10 000*	21 195.616921	1 000*	1 105.532479
MA	53	2.666887	43	2.110139
MA_RS	53	1.421567	43	1.233016
MA_NR	58	2.682151	47	2.061930
MMA1	66	3.332977	70	3.254242
MMA1_RS	64	1.746444	58	1.582230
MMA1_NR	154	8.837334	147	8.037586
MMA2	203	9.234192	157	7.073227
MMA2_RS	203	5.515957	157	4.103546
MMA2_NR	244	10.608097	216	9.966142
MMA3	60	2.884311	55	2.615648
MMA3_RS	60	1.567546	55	1.469893
MMA3_NR	146	10.699405	78	4.494860
MMA4	59	2.871880	57	2.779204
MMA4_RS	59	1.524144	57	1.486978
MMA4_NR	95	5.757624	97	6.133137
Poznámky: * iterační výpočet ukončen před nalezením ε-přesného řešení ** výpočet ukončen s hlášením: „!!! ŘEŠENÍ NELZE NAJÍT !!!“				

Testovací úloha lp_d2q06c

Volba $\tau = 0.999$ se nejeví jako vhodná pro hledání optimálního řešení úlohy lp_d2q06c pomocí metody sledování cesty s dlouhým krokem (NCDK, NCDK_RS a NCDK_NR) a základního primárně-duálního algoritmu (PD, PD_RS a PD_NR). Poměrně „vysoká“ časová náročnost výpočtu u první jmenované vedla k ukončení iteračního výpočtu dříve než bylo ε-přesné řešení nalezeno. Toto je možné, při zachování $\tau = 0.999$, napravit volbou $\gamma = 10^{-2}$. V případě aplikace základního primárně-duálního algoritmu dochází k špatné identifikaci (ne)existence optimálního řešení, což se pro PD_RS a PD_NR změní, je-li hodnota K zvýšena na 10^{35}. Co se týče Mehrotrova algoritmu a jeho modifikací, lze prohlásit, že MA, MA_RS, MA_NR, MMA1, MMA1_RS, MMA1_NR, MMA2, MMA2_RS, MMA2_NR dosahují lepších vlastností z hlediska počtu iterací i doby výpočtu pro $\tau = 0.999$, MMA3, MMA3_RS, MMA4, MMA4_RS pro $\tau = 0.9$. MMA3_NR vykazuje nižší počet iterací pro $\tau = 0.9$, ale kratší dobu výpočtu pro $\tau = 0.999$. MMA4_NR udává pro $\tau = 0.9$ i $\tau = 0.999$ stejný počet iterací, ale kratší dobu výpočtu pro vyšší z uvedených hodnot.

Tabulka 4.24: Výsledky pro testovací úlohu lp_d2q06c při různých hodnotách parametru τ

	$\tau = 0.9$		$\tau = 0.999$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	88	155.222846	2 000*	5 512.354690
NCDK_RS	88	18.762213	2 000*	633.353024
NCDK_NR	161	69.490212	2 000*	1 359.667897
PD	84	146.800629	140**	357.764004
PD_RS	84	18.420236	94**	22.182891
PD_NR	92	25.766696	175**	98.787953
MA	48	186.729835	43	163.379608
MA_RS	48	20.730962	43	17.789355
MA_NR	51	28.011797	44	20.814776
MMA1	110	344.789915	96	293.984576
MMA1_RS	96	37.490286	92	35.700138
MMA1_NR	114	54.791494	96	40.905525
MMA2	653	1 874.927178	624	1 804.283557
MMA2_RS	653	302.377782	624	289.994081
MMA2_NR	658	345.960032	627	329.203430
MMA3	59	209.236012	61	208.135541
MMA3_RS	59	25.056311	61	25.033599
MMA3_NR	62	34.394256	65	35.844309
MMA4	53	194.181809	55	193.472417
MMA4_RS	53	23.206499	55	22.756537
MMA4_NR	56	32.571565	56	26.613247
Poznámky: * iterační výpočet ukončen před nalezením ε-přesného řešení ** výpočet ukončen s hlášením: „!!! ŘEŠENÍ NELZE NAJÍT !!!“				

Testovací úloha lp_ship04l

Opět při hledání optimálního řešení úlohy lp_ship04l pomocí metody sledování cesty s dlouhým krokem (NCDK, NCDK_RS a NCDK_NR) a základního primárně-duálního algoritmu (PD, PD_RS a PD_NR) není vhodnou volbou $\tau = 0.999$. Výpočetní problémy lze u první jmenované vyřešit použitím $\gamma = 10^{-2}$, u základního primárně-duálního algoritmu zvýšením hodnoty parametru K. Mehrotrův algoritmus (MA, MA_RS a MA_NR) a jeho modifikace (MMA1, MMA1_RS, MMA1_NR, MMA2, MMA2_RS, MMA2_NR, MMA3, MMA3_RS, MMA3_NR, MMA4, MMA4_RS a MMA4_NR) vykazují pro $\tau = 0.999$ nižší počet iterací i kratší dobu výpočtu než pro $\tau = 0.9$.

Tabulka 4.25: Výsledky pro testovací úlohu lp_ship04l při různých hodnotách parametru τ

	$\tau = 0.9$		$\tau = 0.999$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	58	8.004368	10 000*	1 482.314059
NCDK_RS	58	2.071377	10 000*	419.756568
NCDK_NR	58	1.036704	10 000*	229.684191
PD	55	7.512500	77**	10.395455
PD_RS	55	1.929808	77**	2.418896
PD_NR	55	0.938186	77**	1.155447
MA	29	7.943988	26	7.107833
MA_RS	29	1.992358	26	1.834397
MA_NR	29	0.958045	26	0.822993
MMA1	36	9.884194	30	8.170717
MMA1_RS	37	2.472949	30	2.037025
MMA1_NR	36	1.092687	30	0.926921
MMA2	61	16.406255	53	14.230661
MMA2_RS	61	4.101146	53	3.581329
MMA2_NR	61	1.772260	53	1.550438
MMA3	32	8.649426	25	6.780696
MMA3_RS	32	2.145920	25	1.721176
MMA3_NR	32	0.986550	25	0.795759
MMA4	30	8.128638	27	7.323200
MMA4_RS	30	2.034057	27	1.851077
MMA4_NR	30	0.923815	27	0.909371
Poznámky: * iterační výpočet ukončen před nalezením ε-přesného řešení ** výpočet ukončen s hlášením: „!!! ŘEŠENÍ NELZE NAJÍT !!!“				

Testovací úloha lp_ship04s

Výpočetní problémy, které se objevují při řešení úlohy lp_ship04s metodou sledování cesty s dlouhým krokem (NCDK, NCDK_RS a NCDK_NR) nebo základním primárně-duálním algoritmem (PD, PD_RS a PD_NR), je možné vyřešit stejně jako u předcházející úlohy. Co se týče Mehrotrova algoritmu (MA, MA_RS, a MA_NR) a jeho modifikace č. 1 (MMA1, MMA1_RS a MMA1_NR), jeví se jako výhodnější použít $\tau = 0.999$ než $\tau = 0.9$. Totéž platí pro MMA4_RS. Z hlediska počtu iterací je lhostejné, zda při aplikaci modifikace Mehrotrova algoritmu č. 2 (MMA2, MMA2_RS a MMA2_NR), je použita volba $\tau = 0.9$, nebo $\tau = 0.999$, z hlediska doby výpočtu je pro MMA2 a MMA2_NR lepší nižší z hodnot a pro MMA2_RS vyšší. MMA3, MMA3_RS, MMA3_NR, MMA4 a MMA4_NR vykazují lepší výpočetní vlastnosti, je-li $\tau = 0.9$.

Tabulka 4.26: Výsledky pro testovací úlohu lp_ship04s při různých hodnotách parametru τ

	$\tau = 0.9$		$\tau = 0.999$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	71	5.317426	10 000*	882.115756
NCDK_RS	71	1.703349	10 000*	362.268304
NCDK_NR	75	0.860625	10 000*	217.162909
PD	69	5.106917	72**	5.314085
PD_RS	69	1.617936	72**	1.645442
PD_NR	69	0.786650	72**	0.797391
MA	33	4.912245	26	3.879988
MA_RS	33	1.586894	26	1.247706
MA_NR	33	0.743717	26	0.610083
MMA1	40	5.969333	36	5.374976
MMA1_RS	40	1.915789	37	1.787546
MMA1_NR	40	0.896270	36	0.803026
MMA2	66	9.795041	66	9.797339
MMA2_RS	66	3.125191	66	3.107465
MMA2_NR	66	1.436199	66	1.438857
MMA3	35	5.183459	39	5.786352
MMA3_RS	35	1.675427	39	1.843303
MMA3_NR	35	0.798457	39	0.876287
MMA4	36	5.307760	36	5.311949
MMA4_RS	36	1.725420	36	1.699583
MMA4_NR	36	0.814783	36	0.806767

Poznámky: * iterační výpočet ukončen před nalezením ε-přesného řešení
** výpočet ukončen s hlášením: „!!! ŘEŠENÍ NELZE NAJÍT !!!“

Testovací úloha lp_ship08l

Pro nalezení ε-přesného řešení úlohy lp_ship08l pomocí metody sledování cesty s dlouhým krokem (NCDK, NCDK_RS a NCDK_NR) a základního primárně-duálního algoritmu (PD, PD_RS a PD_NR) platí závěry učiněné pro aplikaci těchto metod na úlohu lp_ship04l. Z hlediska počtu iterací i doby výpočtu je pro Mehrotrův algoritmus (MA, MA_RS a MA_NR) nebo jeho modifikace (MMA1, MMA1_RS, MMA1_NR, MMA2, MMA2_RS, MMA2_NR, MMA3, MMA3_RS, MMA3_NR, MMA4, MMA4_RS a MMA4_NR) výhodnější použít vyšší z uvedených hodnot parametru τ. Jedinou výjimkou je MMA1_RS, která pro $\tau = 0.9$ vykazuje nižší počet iterací i kratší dobu výpočtu.

Tabulka 4.27: Výsledky pro testovací úlohu lp_ship08l při různých hodnotách parametru τ

	$\tau = 0.9$		$\tau = 0.999$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	77	21.371969	10 000*	2 934.994653
NCDK_RS	77	5.344283	10 000*	735.679523
NCDK_NR	77	2.380944	10 000*	507.578003
PD	69	19.025598	87**	23.738599
PD_RS	69	4.715919	85**	5.615637
PD_NR	69	2.093808	90**	2.652367
MA	31	17.142313	27	14.923871
MA_RS	31	4.295941	27	3.733291
MA_NR	31	1.917851	27	1.690505
MMA1	39	21.576432	39	21.561054
MMA1_RS	41	5.644719	42	5.721011
MMA1_NR	39	2.349510	39	2.345907
MMA2	72	39.709467	67	36.867806
MMA2_RS	72	9.778588	67	9.112814
MMA2_NR	72	4.165235	67	3.883929
MMA3	32	17.704319	31	17.177723
MMA3_RS	32	4.442437	31	4.315247
MMA3_NR	32	1.974394	31	1.917313
MMA4	39	21.560303	35	19.346882
MMA4_RS	39	5.350821	35	4.812377
MMA4_NR	39	2.362211	35	2.145139
Poznámky: * iterační výpočet ukončen před nalezením ε-přesného řešení ** výpočet ukončen s hlášením: „!!! ŘEŠENÍ NELZE NAJÍT !!!“				

Testovací úloha lp_ship08s

Závěry učiněné pro úlohu lp_ship04l platí i pro tuto úlohu, jak je patrné ze srovnání tabulek s výsledky.

Tabulka 4.28: Výsledky pro testovací úlohu lp_ship08s při různých hodnotách parametru τ

	$\tau = 0.9$		$\tau = 0.999$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	73	7.859535	10 000*	1 253.795918
NCDK_RS	73	2.841248	10 000*	446.644133
NCDK_NR	73	1.348446	10 000*	299.584244
PD	69	7.396232	73**	7.852923
PD_RS	69	2.625497	78**	2.864598
PD_NR	69	1.227193	100**	1.770709
MA	33	7.070627	23	4.935167
MA_RS	33	2.538275	23	1.787320
MA_NR	33	1.185616	23	0.845928
MMA1	46	9.835511	40	8.538729
MMA1_RS	46	3.553331	40	3.115751
MMA1_NR	46	1.621802	40	1.426142
MMA2	59	12.580133	53	11.295847
MMA2_RS	59	4.547968	53	4.084803
MMA2_NR	59	2.059433	53	1.869362
MMA3	40	8.578224	34	7.271083
MMA3_RS	40	3.090423	34	2.639701
MMA3_NR	40	1.433657	34	1.226703
MMA4	35	7.504646	33	7.083735
MMA4_RS	35	2.704611	33	2.552103
MMA4_NR	35	1.278289	33	1.201453
Poznámky: * iterační výpočet ukončen před nalezením ε-přesného řešení ** výpočet ukončen s hlášením: „!!! ŘEŠENÍ NELZE NAJÍT !!!“				

Testovací úloha lp_ship12l

Pouze pro metodu sledování cesty s dlouhým krokem (NCDK, NCDK_RS a NCDK_NR), základní primárně-duálního algoritmus (PD, PD_RS a PD_NR) a modifikaci Mehrotrova algoritmu č. 1 (MMA1, MMA1_RS a MMA1_NR) je při řešení úlohy lp_ship12l výhodnější použít $\tau = 0.9$. Na řešení výpočetních problémů pro první dvě z uvedených metod viz např. předcházející úloha.

Tabulka 4.29: Výsledky pro testovací úlohu lp_ship12l při různých hodnotách parametru τ

	$\tau = 0.9$		$\tau = 0.999$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	74	22.560795	10 000*	3 523.548612
NCDK_RS	74	6.497408	10 000*	1 169.008463
NCDK_NR	74	2.904813	10 000*	562.724260
PD	65	19.701658	68**	20.872870
PD_RS	65	5.618268	68**	5.643029
PD_NR	65	2.481739	70**	2.663206
MA	32	19.517358	27	16.506484
MA_RS	32	5.594810	27	4.784163
MA_NR	32	2.456912	27	2.113289
MMA1	50	30.538922	51	31.063742
MMA1_RS	50	8.646467	51	8.752667
MMA1_NR	50	3.742413	51	3.794017
MMA2	64	38.878320	64	38.748496
MMA2_RS	64	10.984614	64	10.911851
MMA2_NR	65	4.817151	64	4.729863
MMA3	38	23.233908	37	22.527776
MMA3_RS	38	6.649872	37	6.438662
MMA3_NR	38	2.902072	37	2.824047
MMA4	41	24.877533	40	24.294828
MMA4_RS	41	7.129456	40	6.910955
MMA4_NR	41	3.127272	40	3.061984
Poznámky: * iterační výpočet ukončen před nalezením ε-přesného řešení ** výpočet ukončen s hlášením: „!!! ŘEŠENÍ NELZE NAJÍT !!!“				

Testovací úloha lp_ship12s

Při aplikaci NCDK, NCDK_RS, NCDK_NR, PD, PD_RS, PD_NR, MMA1, MMA4, MMA4_RS a MMA4_NR na úlohu lp_ship12s je z hlediska počtu iterací i doby výpočtu výhodnější použít $\tau = 0.9$. U ostatních metod naopak $\tau = 0.999$.

Tabulka 4.30: Výsledky pro testovací úlohu lp_ship12s při různých hodnotách parametru τ

	$\tau = 0.9$		$\tau = 0.999$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	64	6.762548	10 000*	1 255.285494
NCDK_RS	65	3.045373	10 000*	532.489460
NCDK_NR	64	1.434702	10 000*	413.531898
PD	56	5.882228	71**	7.493824
PD_RS	56	2.560983	71**	3.083635
PD_NR	56	1.217489	71**	1.497233
MA	32	6.715752	27	5.666925
MA_RS	32	2.926204	27	2.468718
MA_NR	32	1.368552	27	1.165184
MMA1	52	10.886814	53	11.079780
MMA1_RS	53	4.837535	51	4.631708
MMA1_NR	52	2.195952	53	2.228277
MMA2	140	28.963169	78	16.145946
MMA2_RS	140	12.362824	78	6.920131
MMA2_NR	140	5.785728	78	3.264434
MMA3	41	8.589559	40	8.394766
MMA3_RS	41	3.797444	40	3.650076
MMA3_NR	41	1.764593	40	1.714223
MMA4	39	8.170335	40	8.377498
MMA4_RS	39	3.614470	40	3.640188
MMA4_NR	39	1.699396	40	1.717311

Poznámky: * iterační výpočet ukončen před nalezením ε-přesného řešení
** výpočet ukončen s hlášením: „!!! ŘEŠENÍ NELZE NAJÍT !!!“

Testovací úloha lpi_bgprtr

NCDK_NR se pro identifikaci neexistence optimálního řešení úlohy lpi_bgprtr nejeví jako vhodná, a to ať již je použita volba $\tau = 0.9$, nebo $\tau = 0.999$. Z hlediska počtu iterací i doby výpočtu je při aplikaci PD_NR, MA_NR, MMA1, MMA1_NR, MMA2_RS, MMA3 a MMA3_NR výhodnější za hodnotu τ zvolit 0.9. U ostatních metod je lepší použít $\tau = 0.999$.

Tabulka 4.31: Výsledky pro testovací úlohu lpi_bgprtr při různých hodnotách parametru τ

	$\tau = 0.9$		$\tau = 0.999$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	123	0.266643	120	0.265369
NCDK_RS	102	0.132192	116	0.147436
NCDK_NR	10 000*	30.315618	10 000*	23.512289
PD	66	0.140550	34	0.084346
PD_RS	66	0.088314	59	0.085107
PD_NR	61	0.082092	96	0.104990
MA	15	0.074059	12	0.064241
MA_RS	12	0.055791	12	0.052338
MA_NR	14	0.052461	19	0.063425
MMA1	25	0.110446	32	0.133856
MMA1_RS	22	0.072800	18	0.061789
MMA1_NR	28	0.074325	33	0.082192
MMA2	73	0.278929	68	0.252885
MMA2_RS	60	0.135174	61	0.137301
MMA2_NR	144	0.266795	143	0.263558
MMA3	22	0.101764	25	0.115235
MMA3_RS	23	0.077883	23	0.075479
MMA3_NR	30	0.081369	33	0.089501
MMA4	93	0.502091	76	0.407891
MMA4_RS	57	0.171257	57	0.169522
MMA4_NR	86	0.214531	51	0.141163
Poznámka: * iterační výpočet ukončen předčasně bez rozpoznání neexistence řešení				

Testovací úloha lpi_itest6

K identifikaci neexistence optimálního řešení úlohy lpi_itest6 se NCDK_NR nejeví jako vhodná, a to pro kteroukoli z uvedených hodnot parametru τ. Pouze pro MA, MA_RS, MA_NR a MMA4_NR je výhodnější použít $\tau = 0.9$.

Tabulka 4.32: Výsledky pro testovací úlohu lpi_itest6 při různých hodnotách parametru τ

	$\tau = 0.9$		$\tau = 0.999$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	134	0.177257	128	0.173004
NCDK_RS	134	0.134760	128	0.132355
NCDK_NR	10 000*	8.857725	10 000*	8.913058
PD	80	0.105526	72	0.104750
PD_RS	80	0.083303	72	0.081205
PD_NR	6 603	4.481774	476	0.317812
MA	33	0.094358	35	0.095283
MA_RS	33	0.071124	35	0.078393
MA_NR	33	0.071181	35	0.076461
MMA1	30	0.083318	27	0.081067
MMA1_RS	30	0.068588	27	0.065406
MMA1_NR	30	0.069398	28	0.067261
MMA2	75	0.168241	74	0.165021
MMA2_RS	75	0.127202	74	0.126145
MMA2_NR	255	0.333703	218	0.286735
MMA3	35	0.101071	32	0.092034
MMA3_RS	35	0.079914	32	0.076909
MMA3_NR	35	0.074602	32	0.073141
MMA4	73	0.236150	72	0.216110
MMA4_RS	73	0.167024	72	0.162358
MMA4_NR	227	0.424922	290	0.489152
Poznámky: * iterační výpočet ukončen předčasně bez rozpoznání neexistence řešení				

Zhodnocení metod při změně hodnoty parametru τ

Výpočetní problémy se objevují při řešení úloh metodou sledování cesty s dlouhým krokem (NCDK, NCDK_RS a NCDK_NR) a základním primárně-duálním algoritmem (PD, PD_RS a PD_NR), je-li hodnota τ 0.999. Odtud vyplývá, že volba $\tau = 0.9$ je mnohem výhodnější. Co se týče Mehrotrova algoritmu, je při jeho aplikaci na úlohy, které mají optimální řešení, nalezeno ε-přesné řešení při vyšší hodnotě τ rychleji. Pro modifikace Mehrotrova algoritmu platí, že při řešení některých úloh je vhodnější použít nižší a jiných vyšší hodnotu parametru τ, přičemž při řešení naprosté většiny zde uvedených problémů je lepší volit $\tau = 0.999$.

4.3.6 Výsledky metod při změnách hodnoty parametru δ

Základní volba pro parametr δ byla $\delta = 10$. Změny chování metod vnitřních bodů při změně hodnoty tohoto parametru na hodnotu 1 a na hodnotu 100 je demonstrována výsledky obsaženými v tabulkách níže. Hodnoty ostatních parametrů zůstávají nezměněny, tj. stejné jako v případě základní volby $\left(\gamma = 10^{-3},\ \tau = 0.9,\ \beta = \frac{1}{5},\ K = 10^{15} \text{ a } \varepsilon = 10^{-6}\right)$.

Testovací úloha lp_adlittle

Při řešení úlohy lp_adlittle metodou sledování cesty s dlouhým krokem je nejlepší z hlediska počtu iterací i doby potřebné pro nalezení ε-přesného řešení volit $\delta = 10$. Ačkoli pro $\delta = 100$ je počet iterací stejný, doby výpočtu jsou delší. Volba hodnoty δ na jeho dolní hranici, tj. $\delta = 1$, vykazuje nejvyšší počet iterací.

Tabulka 4.33: Výsledky pro testovací úlohu lp_adlittle při různých hodnotách parametru δ

	$\delta = 1$		$\delta = 10$		$\delta = 100$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	36	0.173639	30	0.166011	30	0.170672
NCDK_RS	36	0.136715	30	0.096201	30	0.113637
NCDK_NR	36	0.112124	30	0.078750	30	0.115976

Testovací úloha lp_afiro

Stejně jako při řešení předcházející úlohy je i při řešení lp_afiro počet iterací potřebný pro nalezení ε-přesného řešení pro $\delta = 10$ a $\delta = 100$ stejný. Doba výpočtu je však nižší pro $\delta = 100$. Volba δ na jeho dolní hranici, tj. $\delta = 1$, se nejeví jako vhodná, neboť ε-přesné řešení je při ní nalezeno s poměrně vysokým počtem iterací, vzhledem k rozměrům matice úlohy $\mathbf{A}$ ale doba výpočtu není „příliš dlouhá“.

Tabulka 4.34: Výsledky pro testovací úlohu lp_afiro při různých hodnotách parametru δ

	$\delta = 1$		$\delta = 10$		$\delta = 100$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	769	1.632598	17	0.061247	17	0.056506
NCDK_RS	769	1.184381	17	0.052555	17	0.052452
NCDK_NR	769	1.064112	17	0.051301	17	0.050882

Testovací úloha lp_agg

Použití NCDK_NR pro určení ε-přesného řešení úlohy lp_agg není vhodné, neboť ani v 10 000. iteraci není nalezeno. Toto neovlivní ani změna hodnoty parametru δ.

Při aplikaci NCDK nebo NCDK_RS je z hlediska počtu iterací hodnota parametru δ (v rámci uvedených hodnot) lhostejná, z hlediska času se jako výhodnější jeví zvolit $\delta = 1$, nebo $\delta = 100$.

Tabulka 4.35: Výsledky pro testovací úlohu lp_agg při různých hodnotách parametru δ

	$\delta = 1$		$\delta = 10$		$\delta = 100$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	150	3.496115	150	5.324811	150	3.440967
NCDK_RS	150	1.945408	150	2.880378	150	1.903255
NCDK_NR	10 000*	404.143702	10 000*	408.421373	10 000*	376.831555
Poznámky: * iterační výpočet ukončen před nalezením ε-přesného řešení						

Testovací úloha lp_d2q06c

Při aplikaci NCDK a NCDK_RS na úlohu lp_d2q06c je počet iterací při uvedených různých hodnotách parametru δ stejný, zatímco doba výpočtu se mění, byť jen mírně. Při použití NCDK_NR je nejvhodnější zvolit $\delta = 100$, kdy je počet iterací nejnižší, a tím i doba výpočtu nejkratší.

Tabulka 4.36: Výsledky pro testovací úlohu lp_d2q06c při různých hodnotách parametru δ

	$\delta = 1$		$\delta = 10$		$\delta = 100$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	88	155.260093	88	155.222846	88	155.358529
NCDK_RS	88	19.030710	88	18.762213	88	19.054485
NCDK_NR	342	191.448927	161	69.490212	115	39.037284

Testovací úloha lp_ship04l

Z údajů uvedených v tabulce 4.37, které se týkají nalezení ε-přesného řešení úlohy lp_ship04l, vyplývá, že počet iterací se se změnou hodnoty parametru δ nemění, s jedinou výjimkou: jedná se o případ, kdy je směr určován pomocí rozšířené soustavy (pro $\delta = 1$ je počet iterací, potřebný pro nalezení ε-přesného řešení, vyšší, a tím i doba výpočtu delší). Doba výpočtu je pro NCDK_RS a NCDK_NR nejkratší, je-li $\delta = 100$, pro NCDK, je-li $\delta = 1$.

Tabulka 4.37: Výsledky pro testovací úlohu lp_ship04l při různých hodnotách parametru δ

	$\delta = 1$		$\delta = 10$		$\delta = 100$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	58	7.847026	58	8.004368	58	7.851018
NCDK_RS	61	2.082404	58	2.071377	58	1.980390
NCDK_NR	58	0.927601	58	1.036704	58	0.897202

Testovací úloha lp_ship04s

Pro určení ε-přesného řešení úlohy lp_ship04s pomocí NCDK_RS nebo NCDK_NR se volba $\delta = 1$, nejeví jako vhodná, neboť ani v 10 000. iteraci není nalezeno. Naopak nejvýhodnější, jak z hlediska počtu iterací, tak i času je použití $\delta = 100$. Toto platí i při aplikaci NCDK.

Tabulka 4.38: Výsledky pro testovací úlohu lp_ship04s při různých hodnotách parametru δ

	$\delta = 1$		$\delta = 10$		$\delta = 100$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	73	5.768917	71	5.317426	71	5.243639
NCDK_RS	10 000*	342.109732	71	1.703349	71	1.691497
NCDK_NR	10 000*	194.174800	75	0.860625	71	0.789375
Poznámky: * iterační výpočet ukončen před nalezením ε-přesného řešení						

Testovací úloha lp_ship08l

Pro určení ε-přesného řešení úlohy lp_ship08l pomocí NCDK_RS se volba $\delta = 1$, nejeví jako vhodná, neboť ani v 10 000. iteraci není nalezeno. Je-li pominuta volba $\delta = 1$ pro NCDK a pro NCDK_RS (počty iterací jsou v těchto případech nejvyšší) je z hlediska počtu iterací lhostejné, zda je hodnota δ 1, 10, nebo 100. Z hlediska času je nejvýhonější pro NCDK_RS a NCDK_NR zvolit hodnotu δ nejvyšší uváděnou a pro NCDK 10.

Tabulka 4.39: Výsledky pro testovací úlohu lp_ship08l při různých hodnotách parametru δ

	$\delta = 1$		$\delta = 10$		$\delta = 100$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	80	22.283679	77	21.371969	77	21.381397
NCDK_RS	10 000*	859.284242	77	5.344283	77	5.343178
NCDK_NR	77	2.373099	77	2.380944	77	2.323519
Poznámky: * iterační výpočet ukončen před nalezením ε-přesného řešení						

Testovací úloha lp_ship08s

Situace při hledání ε-přesného řešení úlohy lp_ship08s se podobá situaci popsané u úlohy lp_ship08s. Nejnižší počty iterací jsou vykazovány pro $\delta = 10$ a $\delta = 100$. Z hlediska doby potřebné k výpočtu je pro NCDK a NCDK_RS nejvýhodnější použít volbu $\delta = 10$ a pro NCDK_NR $\delta = 100$.

Tabulka 4.40: Výsledky pro testovací úlohu lp_ship08s při různých hodnotách parametru δ

	$\delta = 1$		$\delta = 10$		$\delta = 100$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	74	8.018801	73	7.859535	73	7.900674
NCDK_RS	10 000*	515.448697	73	2.841248	73	2.843122
NCDK_NR	74	1.377069	73	1.348446	73	1.335346
Poznámky: * iterační výpočet ukončen před nalezením ε-přesného řešení						

Testovací úloha lp_ship12l

Pro zhodnocení výsledků uvedených v tabulce 4.41 je možné využít analogie s výsledky úlohy lp_ship08s. Přitom z hlediska počtu iterací a současně doby výpočtu je pro NCDK nejvýhodnější použít volbu $\delta = 10$ a pro NCDK_RS a NCDK_NR $\delta = 100$.

Tabulka 4.41: Výsledky pro testovací úlohu lp_ship12l při různých hodnotách parametru δ

	$\delta = 1$		$\delta = 10$		$\delta = 100$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	75	22.953966	74	22.560795	74	22.640985
NCDK_RS	10 000*	1 115.801115	74	6.497408	74	6.490517
NCDK_NR	75	3.035658	74	2.904813	74	2.836809
Poznámky: * iterační výpočet ukončen před nalezením ε-přesného řešení						

Testovací úloha lp_ship12s

Z výsledků uvedených v následující tabulce vylývá, že volba $\delta = 1$ není při hledání ε-přesného řešení úlohy lp_ship12s vhodná, protože ani v 10 000. iteraci není nalezeno. Jako nejvýhodnější se pro NCDK, NCDK i NCDK_RS jeví položit $\delta = 100$, a to, jak z hlediska počtu iterací, tak i doby výpočtu.

Tabulka 4.42: Výsledky pro testovací úlohu lp_ship12s při různých hodnotách parametru δ

	$\delta = 1$		$\delta = 10$		$\delta = 100$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	10 000*	1 251.912829	64	6.762548	64	6.753796
NCDK_RS	10 000*	664.968671	65	3.045373	64	2.978819
NCDK_NR	10 000*	389.464586	64	1.434702	64	1.400417
Poznámky: * iterační výpočet ukončen před nalezením ε-přesného řešení						

Testovací úloha lpi_bgprtr

Použití NCDK_NR pro identifikaci neexistence optimálního řešení úlohy lpi_bgprtr se nejeví jako vhodné, a to se nezmění ani při změně hodnoty parametru δ (neexistence není rozpoznána ani v 10 000. iteraci, ať již je hodnota δ rovna 1, 10, nebo 100). Při aplikaci NCDK nebo NCDK_RS se z hlediska počtu iterací i doby výpočtu jako nejvýhodnější jeví zvolit $\delta = 10$.

Tabulka 4.43: Výsledky pro testovací úlohu lpi_bgprtr při různých hodnotách parametru δ

	$\delta = 1$		$\delta = 10$		$\delta = 100$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	124	0.295871	123	0.266643	123	0.285092
NCDK_RS	104	0.132389	102	0.132192	102	0.178563
NCDK_NR	10 000*	28.251607	10 000*	30.315618	10 000*	32.545891
Poznámky: * iterační výpočet ukončen předčasně bez rozpoznání neexistence řešení						

Testovací úloha lpi_itest6

Závěr týkající se použití NCDK_NR pro identifikaci neexistence optimálního řešení úlohy lpi_itest6, je stejný jako u úlohy lpi_bgprtr. Navíc je zde ale možné říci, že i samotná volba $\delta = 1$ není vhodná, protože neexistence řešení není rozpoznána ani v 10 000. iteraci, jak při aplikaci NCDK_NR, tak NCDK, nebo NCDK_RS. Naopak jako nejvýhodnější se pro NCDK i NCDK_RS jeví zvolit $\delta = 10$.

Tabulka 4.44: Výsledky pro testovací úlohu lpi_itest6 při různých hodnotách parametru δ

	$\delta = 1$		$\delta = 10$		$\delta = 100$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	10 000*	16.014942	134	0.177257	134	0.189907
NCDK_RS	10 000*	14.342955	134	0.134760	134	0.135584
NCDK_NR	10 000*	13.568885	10 000*	8.857725	10 000*	9.127464
Poznámky: * iterační výpočet ukončen předčasně bez rozpoznání neexistence řešení						

Zhodnocení metod při změnách hodnoty parametru δ

Na základě provedeného testování, je možné říci, že je lepší volit hodnotu parametru δ vyšší než je jeho dolní mez. Dále lze poznamenat, že NCDK_NR pro identifikaci neexistence řešení není vhodná a ani změna hodnoty δ to neovlivní.

4.3.7 Výsledků metod při změně hodnoty parametru β

Nyní bude zkoumán vliv změny hodnoty parametru β na chování metod vnitřních bodů. Budou posuzovány výstupy metod získané pro volbu $\beta = \gamma^{\frac{1}{4}}$ (hodnota blízká dolní mezi parametru) a volbu $\beta = \frac{1}{5}$ (hodnota blízká jeho horní mezi). Hodnoty ostatních parametrů zůstávají nezměněny, tj. stejné jako v případě základní volby $\left(\gamma = 10^{-3},\ \tau = 0.9,\ \delta = 10,\ K = 10^{15} \text{ a } \varepsilon = 10^{-6}\right)$.

Testovací úloha lp_adlittle

Změnou hodnoty parametru β z $\gamma^{\frac{1}{4}}$ na $\frac{1}{5}$ se počet iterací potřebný k nalezení ε-přesného řešení úlohy lp_adlittle nemění a doba výpočtu se liší jen mírně. Přitom pro MMA3, MMA3_NR, MMA4 a MMA4_NR je kratší pro hodnotu $\beta = \gamma^{\frac{1}{4}}$ a pro MMA3_RS a MMA4_RS pro hodnotu $\beta = \frac{1}{5}$.

Právě uvedený závěr platí i pro úlohy lp_adlittle, lp_ship04l, lp_ship04s, lp_ship08l, lp_ship08s, lp_ship12l a lp_ship12s. Jelikož jejich tabulky s výsledky aplikací jednotlivých metod vypadají podobně jako pro tuto úlohu, nejsou v této práci uvedeny.

Tabulka 4.45: Výsledky pro testovací úlohu lp_adlittle při různých hodnotách parametru β

	$\beta = \gamma^{\frac{1}{4}}$		$\beta = \frac{1}{5}$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
MMA3	24	0.214880	24	0.216548
MMA3_RS	24	0.129644	24	0.129440
MMA3_NR	24	0.110814	24	0.111119
MMA4	23	0.207021	23	0.210204
MMA4_RS	23	0.126063	23	0.125846
MMA4_NR	23	0.132433	23	0.133754

Testovací úloha lp_afiro

Při hledání ε-přesného řešení úlohy lp_afiro je lhostejné, zda je využita volba $\beta = \gamma^{\frac{1}{4}}$, nebo $\beta = \frac{1}{5}$, neboť počet iterací, potřebný k nalezení ε-přesného řešení, je stejný a doba výpočtu se liší jen mírně. Přitom pro MMA3, MMA3_RS a MMA4 je kratší pro hodnotu $\beta = \gamma^{\frac{1}{4}}$ a pro MMA3_NR, MMA4_RS a MMA4_NR pro hodnotu $\beta = \frac{1}{5}$.

Tabulka 4.46: Výsledky pro testovací úlohu lp_afiro při různých hodnotách parametru β

	$\beta = \gamma^{\frac{1}{4}}$		$\beta = \frac{1}{5}$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
MMA3	13	0.072330	13	0.078076
MMA3_RS	13	0.057899	13	0.058801
MMA3_NR	13	0.057334	13	0.055537
MMA4	13	0.072207	13	0.072370
MMA4_RS	13	0.058380	13	0.057747
MMA4_NR	13	0.057929	13	0.056326

Testovací úloha lp_agg

Pouze při aplikaci MMA3_NR a MMA4_NR se při hledání ε-přesného řešení úlohy lp_agg projeví různá hodnota parametru β na počtu iterací. Ovšem zatímco u MMA3_NR je výhodnější použít hodnotu při dolní mezi parametru, tj. $\beta = \gamma^{\frac{1}{4}}$, v případě MMA4_NR je leší volit β při jeho horní mezi, tj. $\beta = \frac{1}{5}$. Z hlediska doby výpočtu je pro MMA3, MMA3_RS a MMA4_NR vhodnější volba $\beta = \frac{1}{5}$ a pro MMA3_NR, MMA4 a MMA4_RS volba $\beta = \gamma^{\frac{1}{4}}$.

Tabulka 4.47: Výsledky pro testovací úlohu lp_agg při různých hodnotách parametru β

	$\beta = \gamma^{\frac{1}{4}}$		$\beta = \frac{1}{5}$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
MMA3	60	2.944687	60	2.884311
MMA3_RS	60	1.602290	60	1.567546
MMA3_NR	120	8.164903	146	10.699405
MMA4	59	2.824562	59	2.871880
MMA4_RS	59	1.516236	59	1.524144
MMA4_NR	101	6.489539	95	5.757624

Testovací úloha lp_d2q06c

Z hlediska počtu iterací je při aplikaci MMA3, MMA3_RS, MMA4 a MMA4_RS na úlohu lp_d2q06c lhostejné, zda je $\beta = \gamma^{\frac{1}{4}}$, nebo $\beta = \frac{1}{5}$. Z hlediska doby výpočtu je pro MMA3, MMA3_RS a MMA4_RS výhodnější volbou $\beta = \gamma^{\frac{1}{4}}$, zatímco pro MMA4 je to $\beta = \frac{1}{5}$. Při použití MMA3_NR a MMA4_NR je vykazován nižší počet iterací i kratší doba výpočtu pro vyšší z uvedených hodnot β.

Tabulka 4.48: Výsledky pro testovací úlohu lp_d2q06c při různých hodnotách parametru β

	$\beta = \gamma^{\frac{1}{4}}$		$\beta = \frac{1}{5}$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
MMA3	59	207.153917	59	209.236012
MMA3_RS	59	24.891257	59	25.056311
MMA3_NR	63	36.511334	62	34.394256
MMA4	53	194.725981	53	194.181809
MMA4_RS	53	23.122636	53	23.206499
MMA4_NR	57	34.841184	56	32.571565

Testovací úloha lpi_bgprtr

Neexistence řešení úlohy lpi_bgprtr je identifikována, jak pro $\beta = \gamma^{\frac{1}{4}}$, tak pro $\beta = \frac{1}{5}$. Při aplikaci MMA3, MMA3_RS, MMA3_NR nebo MMA4_RS je lhostejné, která z uvedených hodnot parametru β bude použita, protože počet iterací je pro jednotlivé varianty stejný a doba výpočtu se liší jen mírně. V případě MMA4 a MMA4_NR se ale počet iterací i doba výpočtu liší v závislosti na volbě parametru β: při užití MMA4 je počet iterací i doba nižší pro $\beta = \gamma^{\frac{1}{4}}$, zatímco při použití MMA4_NR pro $\beta = \frac{1}{5}$.

Tabulka 4.49: Výsledky pro testovací úlohu lpi_bgprtr při různých hodnotách parametru β

	$\beta = \gamma^{\frac{1}{4}}$		$\beta = \frac{1}{5}$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
MMA3	22	0.101151	22	0.101764
MMA3_RS	23	0.070775	23	0.077883
MMA3_NR	30	0.080686	30	0.081369
MMA4	77	0.415678	93	0.502091
MMA4_RS	57	0.173014	57	0.171257
MMA4_NR	42	0.123079	86	0.214531

Testovací úloha lpi_itest6

Při aplikaci MMA3, MMA3_RS, MMA3_NR, MMA4 a MMA4_RS je neexistence řešení úlohy lpi_itest6 pro $\beta = \frac{1}{5}$ i pro $\beta = \gamma^{\frac{1}{4}}$ rozpoznána ve stejném počtu iterací a odlišnosti v době výpočtu jsou zanedbatelné. V tomto směru MMA4_NR představuje jistou výjimku, neboť vyhodnotí neexistenci optimálního řešení pro nižší hodnotu β v nižším počtu iterací a pro vyšší hodnotu β ve vyšším počtu iterací, tomu odpovídá i doba výpočtu.

Tabulka 4.50: Výsledky pro testovací úlohu lpi_itest6 při různých hodnotách parametru β

	$\beta = \gamma^{\frac{1}{4}}$		$\beta = \frac{1}{5}$	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
MMA3	35	0.105065	35	0.101071
MMA3_RS	35	0.079992	35	0.079914
MMA3_NR	35	0.074798	35	0.074602
MMA4	73	0.226176	73	0.236150
MMA4_RS	73	0.166518	73	0.167024
MMA4_NR	190	0.360180	227	0.424922

Zhodnocení metod při změně hodnoty parametru β

V naprosté většině testovaných úloh lineárního programování je lhostejné, zda je za hodnotu parametru β zvolena hodnota $\gamma^{\frac{1}{4}}$ (hodnota při dolní mezi parametru), nebo $\frac{1}{5}$ (hodmota při horní mezi), neboť počet iterací zůstává stejný a doba výpočtu se liší jen nepatrně.

4.3.8 Výsledky metod při nejlepších z předložených hodnot parametrů

Výsledky aplikací metod na testovací úlohy dosahované při nejlepší volbě parametrů, tj. při takových z výše uvedených hodnot, při nichž jsou počty iterací co možná nejnižší a doba výpočtu co možná nejkratší, jsou představeny v následujících tabulkách[4].

Testovací úloha lp_adlittle

Při aplikaci metod na úlohu lp_adlittle je výhodnější určovat směr pomocí systému normálních rovnic, výjimkou je pouze modifikace Mehrotrova algoritmu č. 4. Nejnižší počet iterací i nejkratší dobu výpočtu vykazuje MA_NR. Vhodné je také řešit úlohu pomocí NCDK_NR, jejíž doba výpočtu je jen nepatrně delší.

[4]Hodnoty v tabulkách neuváděných parametrů jsou: $K = 10^{15}$ a $\varepsilon = 10^{-6}$

Tabulka 4.51: Výsledky pro testovací úlohu lp_adlittle při nejlepší volbě parametrů

	Počet iterací	Doba výpočtu	Hodnoty parametrů
NCDK	30	0.166011	$\gamma = 10^{-4}$,$\tau = 0.9$, $\delta = 10$
NCDK_RS	30	0.096201	$\gamma = 10^{-4}$,$\tau = 0.9$, $\delta = 10$
NCDK_NR	30	0.078750	$\gamma = 10^{-4}$,$\tau = 0.9$, $\delta = 10$
PD	30	0.153832	$\tau = 0.9$
PD_RS	30	0.089242	$\tau = 0.9$
PD_NR	30	0.081807	$\tau = 0.9$
MA	15	0.156794	$\tau = 0.999$
MA_RS	15	0.092538	$\tau = 0.999$
MA_NR	15	0.077692	$\tau = 0.999$
MMA1	19	0.171680	$\gamma = 10^{-3}$, $\tau = 0.999$
MMA1_RS	19	0.108019	$\gamma = 10^{-3}$, $\tau = 0.999$
MMA1_NR	19	0.094728	$\gamma = 10^{-3}$, $\tau = 0.999$
MMA2	19	0.180498	$\gamma = 10^{-3}$, $\tau = 0.999$
MMA2_RS	19	0.109070	$\gamma = 10^{-3}$, $\tau = 0.999$
MMA2_NR	19	0.094962	$\gamma = 10^{-3}$, $\tau = 0.999$
MMA3	19	0.180089	$\gamma = 10^{-3}$, $\tau = 0.999$, $\beta = \frac{1}{5}$
MMA3_RS	19	0.109536	$\gamma = 10^{-3}$, $\tau = 0.999$, $\beta = \frac{1}{5}$
MMA3_NR	19	0.096351	$\gamma = 10^{-3}$, $\tau = 0.999$, $\beta = \frac{1}{5}$
MMA4	19	0.175505	$\gamma = 10^{-3}$, $\tau = 0.999$, $\beta = \frac{1}{5}$
MMA4_RS	19	0.109311	$\gamma = 10^{-3}$, $\tau = 0.999$, $\beta = \frac{1}{5}$
MMA4_NR	19	0.119807	$\gamma = 10^{-3}$, $\tau = 0.999$, $\beta = \frac{1}{5}$

Testovací úloha lp_afiro

Při hledání ε-přesného řešení úlohy lp_afiro je prospěšné určovat směr pomocí systému normálních rovnic, přičemž použití MA_NR se stejně jako pro předcházející úlohu jeví jako nejvýhodnější. Nepatrně vyšší časovou náročnost výpočtu vykazuje MA_RS, a poté PD_NR.

Tabulka 4.52: Výsledky pro testovací úlohu lp_afiro při nejlepší volbě parametrů

	Počet iterací	Doba výpočtu	Hodnoty parametrů
NCDK	17	0.056506	$\gamma = 10^{-3}$, $\tau = 0.9$, $\delta = 100$
NCDK_RS	17	0.052452	$\gamma = 10^{-3}$, $\tau = 0.9$, $\delta = 100$
NCDK_NR	17	0.050882	$\gamma = 10^{-3}$, $\tau = 0.9$, $\delta = 100$
PD	17	0.057937	$\tau = 0.9$
PD_RS	17	0.050548	$\tau = 0.9$
PD_NR	17	0.047165	$\tau = 0.9$
MA	7	0.054598	$\tau = 0.999$
MA_RS	7	0.045982	$\tau = 0.999$
MA_NR	7	0.044463	$\tau = 0.999$
MMA1	9	0.057604	$\gamma = 10^{-3}$, $\tau = 0.999$
MMA1_RS	9	0.047770	$\gamma = 10^{-3}$, $\tau = 0.999$
MMA1_NR	9	0.048791	$\gamma = 10^{-3}$, $\tau = 0.999$
MMA2	10	0.064415	$\gamma = 10^{-3}$, $\tau = 0.999$
MMA2_RS	10	0.052873	$\gamma = 10^{-3}$, $\tau = 0.999$
MMA2_NR	10	0.052371	$\gamma = 10^{-3}$, $\tau = 0.999$, $\beta = \frac{1}{5}$
MMA3	9	0.062845	$\gamma = 10^{-3}$, $\tau = 0.999$, $\beta = \frac{1}{5}$
MMA3_RS	9	0.050760	$\gamma = 10^{-3}$, $\tau = 0.999$, $\beta = \frac{1}{5}$
MMA3_NR	9	0.048580	$\gamma = 10^{-3}$, $\tau = 0.999$, $\beta = \frac{1}{5}$
MMA4	9	0.060498	$\gamma = 10^{-3}$, $\tau = 0.999$, $\beta = \frac{1}{5}$
MMA4_RS	9	0.050528	$\gamma = 10^{-3}$, $\tau = 0.999$, $\beta = \frac{1}{5}$
MMA4_NR	9	0.048811	$\gamma = 10^{-3}$, $\tau = 0.999$, $\beta = \frac{1}{5}$

Testovací úloha lp_agg

Použití systému normálních rovnic pro nalezení směru v úloze lp_agg je v případě všech uvedených metod nevýhodné, neboť oproti přímému výpočtu nebo rozšířenému systému vykazuje vyšší počet iterací i delší dobu výpočtu (v případě PD_NR dokonce selhání). Při řešení úlohy lze doporučit využití jednak rozšířenému systému, jednak Mehrotrova algoritmu, příp. jeho modifikace č. 3.

Tabulka 4.53: Výsledky pro testovací úlohu lp_agg při nejlepší volbě parametrů

	Počet iterací	Doba výpočtu	Hodnoty parametrů
NCDK	148	5.348890	$\gamma = 10^{-4}$, $\tau = 0.9$, $\delta = 10$
NCDK_RS	148	2.857506	$\gamma = 10^{-4}$, $\tau = 0.9$, $\delta = 10$
NCDK_NR	10 000*	408.421373	$\gamma = 10^{-3}$, $\tau = 0.9$, $\delta = 10$
PD	146	4.558318	$\tau = 0.999$
PD_RS	143	1.843072	$\tau = 0.999$
PD_NR	1 000*	1105.532479	$\tau = 0.999$
MA	43	2.110139	$\tau = 0.999$
MA_RS	43	1.233016	$\tau = 0.999$
MA_NR	47	2.061930	$\tau = 0.999$
MMA1	66	3.332977	$\gamma = 10^{-3}$, $\tau = 0.9$
MMA1_RS	58	1.582230	$\gamma = 10^{-3}$, $\tau = 0.999$
MMA1_NR	147	8.037586	$\gamma = 10^{-3}$, $\tau = 0.999$
MMA2	157	7.073227	$\gamma = 10^{-3}$, $\tau = 0.999$
MMA2_RS	157	4.103546	$\gamma = 10^{-3}$, $\tau = 0.999$
MMA2_NR	216	9.966142	$\gamma = 10^{-3}$, $\tau = 0.999$
MMA3	55	2.615648	$\gamma = 10^{-3}$, $\tau = 0.999$, $\beta = \frac{1}{5}$
MMA3_RS	55	1.469893	$\gamma = 10^{-3}$, $\tau = 0.999$, $\beta = \frac{1}{5}$
MMA3_NR	78	4.494860	$\gamma = 10^{-3}$, $\tau = 0,999$, $\beta = \frac{1}{5}$
MMA4	57	2.779204	$\gamma = 10^{-3}$, $\tau = 0,999$, $\beta = \frac{1}{5}$
MMA4_RS	57	1.486978	$\gamma = 10^{-3}$, $\tau = 0,999$, $\beta = \frac{1}{5}$
MMA4_NR	97	6.133137	$\gamma = 10^{-3}$, $\tau = 0,999$, $\beta = \frac{1}{5}$
Poznámka: * iterační výpočet ukončen před nalezením ε-přesného řešení			

Testovací úloha lp_d2q06c

Z hlediska počtu iterací nelze použití systému normálních rovnic pro určení směru při řešení úlohy lp_d2q06c doporučit, zato aplikaci rozšířeného systému ano (i z časového hlediska). Nejvýhodnější je pro nalezení ε-přesného řešení využít MA_RS.

Tabulka 4.54: Výsledky pro testovací úlohu lp_d2q06c při nejlepší volbě parametrů

	Počet iterací	Doba výpočtu	Hodnoty parametrů
NCDK	84	146.411314	$\gamma = 10^{-4}$, $\tau = 0.9$, $\delta = 10$
NCDK_RS	84	18.107377	$\gamma = 10^{-4}$, $\tau = 0.9$, $\delta = 10$
NCDK_NR	108	31.686947	$\gamma = 10^{-2}$, $\tau = 0.9$, $\delta = 10$
PD	84	146.800629	$\tau = 0.9$
PD_RS	84	18.420236	$\tau = 0.9$
PD_NR	92	25.766696	$\tau = 0.9$
MA	43	163.379608	$\tau = 0.999$
MA_RS	43	17.789355	$\tau = 0.999$
MA_NR	44	20.814776	$\tau = 0.999$
MMA1	92	294.192246	$\gamma = 10^{-2}$, $\tau = 0.9$
MMA1_RS	89	35.211158	$\gamma = 10^{-2}$, $\tau = 0.9$
MMA1_NR	96	40.905525	$\gamma = 10^{-3}$, $\tau = 0.999$
MMA2	248	764.478079	$\gamma = 10^{-2}$, $\tau = 0.9$
MMA2_RS	248	109.021017	$\gamma = 10^{-2}$, $\tau = 0.9$
MMA2_NR	255	129.972550	$\gamma = 10^{-2}$, $\tau = 0.9$
MMA3	55	198.820246	$\gamma = 10^{-4}$, $\tau = 0.9$, $\beta = \frac{1}{5}$
MMA3_RS	55	23.397161	$\gamma = 10^{-4}$, $\tau = 0.9$, $\beta = \frac{1}{5}$
MMA3_NR	59	35.219917	$\gamma = 10^{-4}$, $\tau = 0.9$, $\beta = \frac{1}{5}$
MMA4	50	189.581331	$\gamma = 10^{-4}$, $\tau = 0.9$, $\beta = \frac{1}{5}$
MMA4_RS	50	22.003258	$\gamma = 10^{-4}$, $\tau = 0.9$, $\beta = \frac{1}{5}$
MMA4_NR	53	31.661040	$\gamma = 10^{-4}$, $\tau = 0.9$, $\beta = \frac{1}{5}$

Testovací úloha lp_ship04l

Nejlepších výsledků při řešení úlohy lp_ship04l dosahuje MA_NR, příp. MMA3_NR. Výhodné je použít systém normálních rovnic pro určení směru. Totéž platí pro úlohu lp_ship08l, byť pro jiné hodnoty parametrů.

Tabulka 4.55: Výsledky pro testovací úlohu lp_ship04l při nejlepší volbě parametrů

	Počet iterací	Doba výpočtu	Hodnoty parametrů
NCDK	55	7.532204	$\gamma = 10^{-4}$, $\tau = 0.9$, $\delta = 10$
NCDK_RS	55	1.948801	$\gamma = 10^{-4}$, $\tau = 0.9$, $\delta = 10$
NCDK_NR	55	0.963009	$\gamma = 10^{-4}$, $\tau = 0.9$, $\delta = 10$
PD	55	7.512500	$\tau = 0.9$
PD_RS	55	1.929808	$\tau = 0.9$
PD_NR	55	0.938186	$\tau = 0.9$
MA	26	7.107833	$\gamma = 10^{-3}$, $\tau = 0.999$
MA_RS	26	1.834397	$\gamma = 10^{-3}$, $\tau = 0.999$
MA_NR	26	0.822993	$\gamma = 10^{-3}$, $\tau = 0.999$
MMA1	30	8.170717	$\gamma = 10^{-3}$, $\tau = 0.999$
MMA1_RS	30	2.037025	$\gamma = 10^{-3}$, $\tau = 0.999$
MMA1_NR	30	0.926921	$\gamma = 10^{-3}$, $\tau = 0.999$
MMA2	51	13.732051	$\gamma = 10^{-4}$, $\tau = 0.9$
MMA2_RS	51	3.431183	$\gamma = 10^{-4}$, $\tau = 0.9$
MMA2_NR	51	1.495419	$\gamma = 10^{-4}$, $\tau = 0.9$
MMA3	25	6.780696	$\gamma = 10^{-3}$, $\tau = 0.999$, $\beta = \frac{1}{5}$
MMA3_RS	25	1.721176	$\gamma = 10^{-3}$, $\tau = 0.999$, $\beta = \frac{1}{5}$
MMA3_NR	25	0.795759	$\gamma = 10^{-3}$, $\tau = 0.999$, $\beta = \frac{1}{5}$
MMA4	27	7.323200	$\gamma = 10^{-3}$, $\tau = 0.999$, $\beta = \frac{1}{5}$
MMA4_RS	27	1.851077	$\gamma = 10^{-3}$, $\tau = 0.999$, $\beta = \frac{1}{5}$
MMA4_NR	27	0.909371	$\gamma = 10^{-3}$, $\tau = 0.999$, $\beta = \frac{1}{5}$

Testovací úloha lp_ship04s

Nejlepší výpočetní vlastnosti při hledání řešení úlohy lp_ship04s vykazuje MA_NR, popř. pak NCDK_NR, MMA3_NR a MMA4_NR. Opět se zde objevuje výhoda použití systému normálních rovnic pro určení směru.

Tabulka 4.56: Výsledky pro testovací úlohu lp_ship04s při nejlepší volbě parametrů

	Počet iterací	Doba výpočtu	Hodnoty parametrů
NCDK	69	5.156197	$\gamma = 10^{-4}$, $\tau = 0.9$, $\delta = 10$
NCDK_RS	69	1.663665	$\gamma = 10^{-4}$, $\tau = 0.9$, $\delta = 10$
NCDK_NR	71	0.789375	$\gamma = 10^{-3}$, $\tau = 0.9$, $\delta = 100$
PD	37	5.517629	$\tau = 0.9$
PD_RS	40	1.915789	$\tau = 0.9$
PD_NR	37	0.851524	$\tau = 0.9$
MA	26	3.879988	$\tau = 0.999$
MA_RS	26	1.247706	$\tau = 0.999$
MA_NR	26	0.610083	$\tau = 0.999$
MMA1	36	5.374976	$\gamma = 10^{-3}$, $\tau = 0.999$
MMA1_RS	37	1.787546	$\gamma = 10^{-3}$, $\tau = 0.999$
MMA1_NR	36	0.803026	$\gamma = 10^{-3}$, $\tau = 0.999$
MMA2	62	9.198825	$\gamma = 10^{-4}$, $\tau = 0.9$
MMA2_RS	62	2.954670	$\gamma = 10^{-4}$, $\tau = 0.9$
MMA2_NR	62	1.340311	$\gamma = 10^{-4}$, $\tau = 0.9$
MMA3	35	5.176251	$\gamma = 10^{-4}$, $\tau = 0.9$, $\beta = \frac{1}{5}$
MMA3_RS	35	1.666030	$\gamma = 10^{-4}$, $\tau = 0.9$, $\beta = \frac{1}{5}$
MMA3_NR	35	0.790579	$\gamma = 10^{-3}$, $\tau = 0.9$, $\beta = \gamma^{\frac{1}{4}}$
MMA4	36	5.289093	$\gamma = 10^{-3}$, $\tau = 0.9$, $\beta = \gamma^{\frac{1}{4}}$,
MMA4_RS	36	1.699583	$\gamma = 10^{-3}$, $\tau = 0.999$, $\beta = \frac{1}{5}$
MMA4_NR	36	0.806767	$\gamma = 10^{-3}$, $\tau = 0.999$, $\beta = \frac{1}{5}$

Testovací úloha lp_ship08s

Pro řešení úlohy lp_ship08s lze doporučit MA_NR, dále pak MMA4_NR, MMA3_NR a PD_NR.

Tabulka 4.57: Výsledky pro testovací úlohu lp_ship08s při nejlepší volbě parametrů

	Počet iterací	Doba výpočtu	Hodnoty parametrů
NCDK	71	7.644184	$\gamma = 10^{-4}$, $\tau = 0.9$, $\delta = 10$
NCDK_RS	72	2.793137	$\gamma = 10^{-4}$, $\tau = 0.9$, $\delta = 10$
NCDK_NR	71	1.313260	$\gamma = 10^{-4}$, $\tau = 0.9$, $\delta = 100$
PD	69	7.396232	$\tau = 0.9$
PD_RS	69	2.625497	$\tau = 0.9$
PD_NR	69	1.227193	$\tau = 0.9$
MA	23	4.935167	$\tau = 0.999$
MA_RS	23	1.787320	$\tau = 0.999$
MA_NR	23	0.845928	$\tau = 0.999$
MMA1	40	8.538729	$\gamma = 10^{-3}$, $\tau = 0.999$
MMA1_RS	40	3.115751	$\gamma = 10^{-3}$, $\tau = 0.999$
MMA1_NR	40	1.426142	$\gamma = 10^{-3}$, $\tau = 0.999$
MMA2	53	11.295847	$\gamma = 10^{-3}$, $\tau = 0.999$
MMA2_RS	53	4.084803	$\gamma = 10^{-3}$, $\tau = 0.999$
MMA2_NR	53	1.869362	$\gamma = 10^{-3}$, $\tau = 0.999$
MMA3	34	7.271083	$\gamma = 10^{-3}$, $\tau = 0.999$, $\beta = \frac{1}{5}$
MMA3_RS	34	2.639701	$\gamma = 10^{-3}$, $\tau = 0.999$, $\beta = \frac{1}{5}$
MMA3_NR	34	1.226703	$\gamma = 10^{-3}$, $\tau = 0.999$, $\beta = \frac{1}{5}$
MMA4	33	7.083735	$\gamma = 10^{-3}$, $\tau = 0.999$, $\beta = \frac{1}{5}$
MMA4_RS	33	2.552103	$\gamma = 10^{-3}$, $\tau = 0.999$, $\beta = \frac{1}{5}$
MMA4_NR	33	1.201453	$\gamma = 10^{-3}$, $\tau = 0.999$, $\beta = \frac{1}{5}$

Testovací úloha lp_ship12l

Při řešení úlohy lp_ship12l je výhodné určovat směr pomocí systému normálních rovnic, přitom jako nejvýhodnější se jeví využití MA_NR, PD_NR, příp. NCDK_NR. Totéž platí i pro úlohu lp_ship12s[5]

[5]Tabulka výsledků pro úlohu lp_ship12s není uváděna.

Tabulka 4.58: Výsledky pro testovací úlohu lp_ship12l při nejlepší volbě parametrů

	Počet iterací	Doba výpočtu	Hodnoty parametrů
NCDK	70	21.328968	$\gamma = 10^{-4}$, $\tau = 0.9$, $\delta = 10$
NCDK_RS	70	6.138041	$\gamma = 10^{-4}$, $\tau = 0.9$, $\delta = 10$
NCDK_NR	70	2.743928	$\gamma = 10^{-4}$, $\tau = 0.9$, $\delta = 10$
PD	65	19.701658	$\tau = 0.9$
PD_RS	65	5.618268	$\tau = 0.9$
PD_NR	65	2.481739	$\tau = 0.9$
MA	27	16.506484	$\tau = 0.999$
MA_RS	27	4.784163	$\tau = 0.999$
MA_NR	27	2.113289	$\tau = 0.999$
MMA1	50	30.527428	$\gamma = 10^{-2}$, $\tau = 0.9$
MMA1_RS	50	8.646467	$\gamma = 10^{-3}$, $\tau = 0.9$
MMA1_NR	50	3.714473	$\gamma = 10^{-4}$, $\tau = 0.9$
MMA2	59	35.796460	$\gamma = 10^{-2}$, $\tau = 0.9$
MMA2_RS	59	10.156778	$\gamma = 10^{-2}$, $\tau = 0.9$
MMA2_NR	59	4.401088	$\gamma = 10^{-2}$, $\tau = 0.9$
MMA3	37	22.527776	$\gamma = 10^{-3}$, $\tau = 0.999$, $\beta = \frac{1}{5}$
MMA3_RS	37	6.438662	$\gamma = 10^{-3}$, $\tau = 0.999$, $\beta = \frac{1}{5}$
MMA3_NR	37	2.824047	$\gamma = 10^{-3}$, $\tau = 0.999$, $\beta = \frac{1}{5}$
MMA4	40	24.275351	$\gamma = 10^{-4}$, $\tau = 0.9$, $\beta = \frac{1}{5}$
MMA4_RS	40	6.910955	$\gamma = 10^{-3}$, $\tau = 0.999$, $\beta = \frac{1}{5}$
MMA4_NR	40	3.056270	$\gamma = 10^{-4}$, $\tau = 0.9$, $\beta = \frac{1}{5}$

Testovací úloha lpi_bgprtr

Nejrychleji odhalí neexistenci optimálního řešení úlohy lpi_bgprtr MA_RS, a to s nejnižším počtem iterací. O něco delší dobu k tomu budou potřebovat MA_NR a MMA1_RS.

Tabulka 4.59: Výsledky pro testovací úlohu lpi_bgprtr při základní volbě parametrů

	Počet iterací	Doba výpočtu	Hodnoty parametrů
NCDK	120	0.265369	$\gamma = 10^{-3}$, $\tau = 0.999$, $\delta = 10$
NCDK_RS	102	0.130789	$\gamma = 10^{-4}$, $\tau = 0.9$, $\delta = 10$
NCDK_NR	10 000*	27.168741	$\gamma = 10^{-2}$, $\tau = 0.9$, $\delta = 10$
PD	34	0.084346	$\tau = 0.999$
PD_RS	59	0.085107	$\tau = 0.999$
PD_NR	61	0.082092	$\tau = 0.9$
MA	12	0.064241	$\tau = 0.999$
MA_RS	12	0.052338	$\tau = 0.999$
MA_NR	14	0.052461	$\tau = 0.9$
MMA1	24	0.101547	$\gamma = 10^{-2}$, $\tau = 0.9$
MMA1_RS	17	0.060244	$\gamma = 10^{-4}$, $\tau = 0.9$
MMA1_NR	27	0.075553	$\gamma = 10^{-4}$, $\tau = 0.9$
MMA2	68	0.252885	$\gamma = 10^{-3}$, $\tau = 0.999$
MMA2_RS	59	0.133402	$\gamma = 10^{-4}$, $\tau = 0.9$
MMA2_NR	96	0.187693	$\gamma = 10^{-2}$, $\tau = 0.9$
MMA3	22	0.101127	$\gamma = 10^{-4}$, $\tau = 0.9$, $\beta = \frac{1}{5}$
MMA3_RS	23	0.070775	$\gamma = 10^{-3}$,$\tau = 0.9$, $\beta = \gamma^{\frac{1}{4}}$
MMA3_NR	26	0.076771	$\gamma = 10^{-2}$, $\tau = 0.9$, $\beta = \frac{1}{5}$
MMA4	75	0.399489	$\gamma = 10^{-2}$, $\tau = 0.9$, $\beta = \frac{1}{5}$
MMA4_RS	56	0.170899	$\gamma = 10^{-4}$, $\tau = 0.9$, $\beta = \frac{1}{5}$
MMA4_NR	42	0.123079	$\gamma = 10^{-3}$, $\tau = 0.9$, $\beta = \gamma^{\frac{1}{4}}$
Poznámka: * iterační výpočet ukončen bez rozpoznání neexistence řešení			

Testovací úloha lpi_itest6

Nejvýhodnější je pro identifikace neexistence optimálního řešení úlohy lpi_itest6 použít MMA1_RS, MMA3_NR nebo MMA1_NR.

Tabulka 4.60: Výsledky pro testovací úlohu lpi_itest6 při nejlepší volbě parametrů

	Počet iterací	Doba výpočtu	Hodnoty parametrů
NCDK	128	0.173004	$\gamma = 10^{-3}$, $\tau = 0.999$, $\delta = 10$
NCDK_RS	128	0.132355	$\gamma = 10^{-3}$, $\tau = 0.999$, $\delta = 10$
NCDK_NR	10 000*	8.818288	$\gamma = 10^{-2}$, $\tau = 0.9$, $\delta = 10$
PD	72	0.104750	$\tau = 0.999$
PD_RS	72	0.081205	$\tau = 0.999$
PD_NR	476	0.317812	$\tau = 0.999$
MA	33	0.094358	$\tau = 0.9$
MA_RS	33	0.071124	$\tau = 0.9$
MA_NR	33	0.071181	$\tau = 0.9$
MMA1	27	0.081067	$\gamma = 10^{-3}$, $\tau = 0.999$
MMA1_RS	27	0.065406	$\gamma = 10^{-3}$, $\tau = 0.999$
MMA1_NR	28	0.067261	$\gamma = 10^{-3}$, $\tau = 0.999$
MMA2	74	0.165021	$\gamma = 10^{-3}$, $\tau = 0.999$
MMA2_RS	74	0.123820	$\gamma = 10^{-4}$, $\tau = 0.9$
MMA2_NR	177	0.238664	$\gamma = 10^{-4}$, $\tau = 0.9$
MMA3	26	0.085473	$\gamma = 10^{-2}$, $\tau = 0.9$, $\beta = \frac{1}{5}$
MMA3_RS	26	0.068757	$\gamma = 10^{-2}$, $\tau = 0.9$, $\beta = \frac{1}{5}$
MMA3_NR	26	0.066271	$\gamma = 10^{-2}$, $\tau = 0.9$, $\beta = \frac{1}{5}$
MMA4	71	0.219904	$\gamma = 10^{-4}$, $\tau = 0.9$, $\beta = \frac{1}{5}$
MMA4_RS	71	0.162726	$\gamma = 10^{-4}$, $\tau = 0.9$, $\beta = \frac{1}{5}$
MMA4_NR	190	0.360180	$\gamma = 10^{-3}$, $\tau = 0.9$, $\beta = \gamma^{\frac{1}{4}}$
Poznámky: * iterační výpočet ukončen předčasně bez rozpoznání neexistence řešení			

Nejvýhodnější je pro identifikace neexistence optimálního řešení úlohy lpi_itest6 použít MMA1_RS, MMA3_NR nebo MMA1_NR.

Zhodnocení metod při užití nejlepší volby parametrů

Pro nalezení ε-přesného řešení lze doporučit Mehrotrův algoritmus, přičemž v mnoha úlohách se jako výhodné jeví určovat směr pomocí systému normálních rovnic. Pro identifikaci neexistence řešení je však tato strategie špatnou volbou. Rozeznání neexistence řešení je spolehlivé při použití rozšířeného systému a současně rychleší než v případě přímého výpočtu soustavy pro stanovení směru..

4.3.9 Adaptivní volba parametrů τ a δ

Při testování různých hodnot parametrů se objevuje problém spočívající v selhávání algoritmů při určitých „nevhodných“ volbách hodnot parametrů a jejich kombinacích. Do jisté míry může být řešením adaptivní volba hodnot parametrů τ a δ. K výpočtu hodnot parametrů v k-té iteraci jsou v této práci využity následující vztahy:

$$\begin{aligned} \tau_k &= max\left\{0.9, 1-\left(\mathbf{x}^k\right)^{\mathrm{T}}\mathbf{s}^k\right\} \\ \delta_k &= 1+(k-1)\,\frac{n}{m} \end{aligned}$$

Dále je demonstrována funkčnost algoritmů jednotlivých metod v případě použití takovýchto voleb, jsou-li hodnoty ostatních parametrů ponechány stejné jako při základní volbě, tj. $\gamma = 10^{-3}$, $\beta = \frac{1}{5}$, $K = 10^{15}$ a $\varepsilon = 10^{-6}$, je demonstrována na několika úlohách.

Testovací úloha lp_adlittle

Adaptivní výpočet parametrů τ a δ přináší při aplikaci metody sledování cesty s dlouhým krokem (NCDK, NCDK_RS a NCDK_NR) a základního primárně-duálního algoritmu (PD, PD_RS a PD_NR) na úlohu lp_adlittle nižší počet iterací než při použití nejlepší volby parametrů. K zkrácení doby výpočtu dochází, jen v případě, že směr není určován pomocí systému normálních rovnic. Přesto právě PD_NR a NCDK_NR (s adaptivní volbou parametrů) vykazují nejkratší dobu výpočtu. Využití adaptivní volby u Mehrotrova algoritmu (MA, MA_RS a MA_NR) a jeho modifikací (MMA1, MMA1_RS, MMA1_NR, MMA2, MMA2_RS, MMA2_NR, MMA3, MMA3_RS, MMA3_NR, MMA4, MMA4_RS a MMA4_NR) naproti tomu znamená vyšší počet iterací i delší dobu výpočtu.

Tabulka 4.61: Výsledky pro testovací úlohu lp_adlittle při adaptivní a nejlepší volbě parametrů

	Adaptivní volba		Nejlepší volba	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	29	0.145067	30	0.166011
NCDK_RS	29	0.118233	30	0.096201
NCDK_NR	29	0.080223	30	0.078750
PD	27	0.163901	30	0.153832
PD_RS	27	0.090781	30	0.089242
PD_NR	27	0.071862	30	0.081807
MA	19	0.162974	15	0.156794
MA_RS	19	0.102263	15	0.092538
MA_NR	19	0.083608	15	0.077692
MMA1	22	0.185248	19	0.171680
MMA1_RS	22	0.116154	19	0.108019
MMA1_NR	22	0.094609	19	0.094728
MMA2	21	0.185816	19	0.180498
MMA2_RS	21	0.112575	19	0.109070
MMA2_NR	21	0.092448	19	0.094962
MMA3	22	0.186053	19	0.180089
MMA3_RS	22	0.116599	19	0.109536
MMA3_NR	22	0.096418	19	0.096351
MMA4	21	0.184166	19	0.175505
MMA4_RS	21	0.113982	19	0.109311
MMA4_NR	21	0.092281	19	0.119807

Testovací úloha lp_afiro

Při použití adaptivního výpočtu parametrů τ a δ metody vykazují kratší dobu výpočtu než při nejlepších hodnotách všech parametrů s jedinou výjimkou, kterou představuje MMA1. Přitom nejkratší dobu výpočtu udávají PD_RS a PD_NR. Počet iterací je stejně jako v přecházejícím případě u metody sledování cesty s dlouhým krokem (NCDK, NCDK_RS a NCDK_NR) a základního primárně-duálního algoritmu (PD, PD_RS a PD_NR) nižší a u Mehrotrova algoritmu (MA, MA_RS a MA_NR) a jeho modifikací vyšší (MMA1, MMA1_RS, MMA1_NR, MMA2, MMA2_RS, MMA2_NR, MMA3, MMA3_RS, MMA3_NR, MMA4, MMA4_RS a MMA4_NR).

Tabulka 4.62: Výsledky pro testovací úlohu lp_afiro při adaptivní a nejlepší volbě parametrů

	Adaptivní volba		Nejlepší volba	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	16	0.053933	17	0.056506
NCDK_RS	16	0.047295	17	0.052452
NCDK_NR	16	0.044947	17	0.050882
PD	15	0.047992	17	0.057937
PD_RS	15	0.040429	17	0.050548
PD_NR	15	0.038910	17	0.047165
MA	10	0.053922	7	0.054598
MA_RS	10	0.044416	7	0.045982
MA_NR	10	0.041277	7	0.044463
MMA1	11	0.057881	9	0.057604
MMA1_RS	11	0.046855	9	0.047770
MMA1_NR	11	0.043835	9	0.048791
MMA2	11	0.058266	10	0.064415
MMA2_RS	11	0.047453	10	0.052873
MMA2_NR	11	0.044952	10	0.052371
MMA3	11	0.057648	9	0.062845
MMA3_RS	11	0.047063	9	0.050760
MMA3_NR	11	0.043495	9	0.048580
MMA4	11	0.058218	9	0.060498
MMA4_RS	11	0.047505	9	0.050528
MMA4_NR	11	0.046563	9	0.048811

Testovací úloha lp_agg

Adaptivní volba parametrů τ a δ přináší při řešení úlohy lp_agg pomocí NCDK_NR značné zrychlení konvergence. Nižší počet iterací i kratší doba výpočtu než při nejlepší volbě parametrů je vykazována MMA1 a MMA4_NR, stený počet iterací, ale kratší doba výpočtu pak PD, MMA4 a MMA4_RS. V rámci metod užívajících adaptivní volby parametrů je ε-přesné řešení nalezeno v nejkratším čase Mehrotrovým algoritmem určující směr pomocí rozšířeného systému (MA_RS). Výpočetní problémy při aplikaci PD_NR však adaptivní volbou nejsou vyřešeny.

Tabulka 4.63: Výsledky pro testovací úlohu lp_agg při adaptivní a nejlepší volbě parametrů

	Adaptivní volba		Nejlepší volba	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	149	3.936525	148	5.348890
NCDK_RS	149	2.200588	148	2.857506
NCDK_NR	767	25.068570	10 000*	408.421373
PD	146	3.503738	146	4.558318
PD_RS	146	1.904375	143	1.843072
PD_NR	10 000*	339.992502	1 000*	1 105.532479
MA	51	2.649574	43	2.110139
MA_RS	51	1.279805	43	1.233016
MA_NR	54	2.294645	47	2.061930
MMA1	63	2.852654	66	3.332977
MMA1_RS	63	1.596007	58	1.582230
MMA1_NR	155	8.566310	147	8.037586
MMA2	201	8.963989	157	7.073227
MMA2_RS	201	5.444549	157	4.103546
MMA2_NR	238	10.167120	216	9.966142
MMA3	57	2.597801	55	2.615648
MMA3_RS	57	1.407756	55	1.469893
MMA3_NR	104	6.754257	78	4.494860
MMA4	57	2.598424	57	2.779204
MMA4_RS	57	1.396480	57	1.486978
MMA4_NR	72	3.767182	97	6.133137
Poznámky: * iterační výpočet ukončen před nalezením ε-přesného řešení				

Testovací úloha lp_d2q06c

Použití adaptivního výpočtu hodnot parametrů τ a δ oproti nejlepší volbě parametrů se jeví jako výhodnější, a to jak z hlediska počtu iterací i doby výpočtu, je-li úloha lp_d2q06c řešena pomocí NCDK_NR. Z hlediska jen doby výpočtu je to pak při aplikaci MA_NR, MMA3, MMA4 a MMA4_RS. Nejkratší dobu výpočtu vykazují (při užití adaptivní volby parametrů τ a δ) NCDK_RS a MA_RS. Problém se špatnou identifikací (ne)existence optimálního řešení, který vzniká při použití PD, je možné vyřešit zvýšením hodnoty parametru K na hodnotu 10^{20} (počet iterací je potom 129).

Tabulka 4.64: Výsledky pro testovací úlohu lp_d2q06c při adaptivní a nejlepší volbě parametrů

	Adaptivní volba		Nejlepší volba	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	88	155.033684	84	146.411314
NCDK_RS	88	18.933593	84	18.107377
NCDK_NR	102	30.555232	108	31.686947
PD	123*	255.875716	84	146.800629
PD_RS	102	22.981910	84	18.420236
PD_NR	126	45.925148	92	25.766696
MA	45	169.589453	43	163.379608
MA_RS	45	18.700394	43	17.789355
MA_NR	45	20.054383	44	20.814776
MMA1	108	333.568536	92	294.192246
MMA1_RS	108	41.759438	89	35.211158
MMA1_NR	116	57.823486	96	40.905525
MMA2	650	1 857.869887	248	764.478079
MMA2_RS	650	296.662845	248	109.021017
MMA2_NR	650	327.897972	255	129.972550
MMA3	57	195.204678	55	198.820246
MMA3_RS	57	23.429292	55	23.397161
MMA3_NR	65	42.198727	59	35.219917
MMA4	51	182.997516	50	189.581331
MMA4_RS	51	21.766631	50	22.003258
MMA4_NR	60	42.551683	53	31.661040
Poznámky: * výpočet ukončen s hlášením: „!!! ŘEŠENÍ NELZE NAJÍT !!!“				

Testovací úloha lp_ship08l

Použití adaptivní volby parametrů τ a δ se jeví jako výhodné při aplikaci PD, PD_RS, PD_NR, MMA1, MMA1_RS, MMA1_NR, MMA3, MMA3_RS a MMA3_NR na úlohu lp_ship08l, neboť ε-přesné řešení je potom nalezeno v nižším počtu iterací a doba výpočtu je kratší než při uvedené nejlepší volbě parametrů. Nejkratší dobu výpočtu při uplatnění adaptivní volby ale udávají MA_NR a MMA3_NR.

Tabulka 4.65: Výsledky pro testovací úlohu lp_ship08l při adaptivní a nejlepší volbě parametrů

	Adaptivní volba		Nejlepší volba	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	75	20.821966	74	20.508674
NCDK_RS	79	5.480612	74	5.141981
NCDK_NR	75	2.307808	74	2.297377
PD	66	18.189264	69	19.025598
PD_RS	66	4.519597	69	4.715919
PD_NR	66	1.992920	69	2.093808
MA	29	16.034007	27	14.923871
MA_RS	29	4.024170	27	3.733291
MA_NR	29	1.795444	27	1.690505
MMA1	37	20.689422	39	21.561054
MMA1_RS	37	5.127265	41	5.644719
MMA1_NR	37	2.247043	39	2.345138
MMA2	69	38.318339	67	36.867806
MMA2_RS	69	9.725215	67	9.112814
MMA2_NR	69	3.983920	67	3.883929
MMA3	30	16.609512	31	17.177723
MMA3_RS	30	4.199020	31	4.315247
MMA3_NR	30	1.859566	31	1.917313
MMA4	37	20.455025	35	19.346882
MMA4_RS	37	5.477100	35	4.812377
MMA4_NR	37	2.241359	35	2.145139

Testovací úloha lpi_bgprtr

Použít adaptivní volbu τ a δ se jeví jako výhodné při aplikaci MMA3_RS na úlohu lpi_bgprtr, která potom vykazuje stený počet iterací a kratší dobu výpočtu než při nejlepší pevné volbě parametrů. Stejný počet iterací pro adaptivní i neadaptivní volbu udávají rovněž PD_NR, MA_RS, MA_NR a MMA3, ovšem doba výpočtu je delší. Kratší dobu výpočtu, ale vyšší počet iterací uvádí MMA1_NR, MMA2_RS, MMA3_NR, MMA4_RS a MMA4_NR. Nejrychleji při využití adaptivní volby parametrů odhalí neexistenci optimálního řešení MA_NR a MA_RS. Ani adaptivní výpočet hodnot parametrů τ a δ nemá v případě NCDK_NR za následek rozpoznání neexistence řešení úlohy lpi_bgprtr do 10 000. iterace. Tento problém je možné vyřešit volbou $K = 10^6$ (neexistence řešení rozpoznána v 70. iteraci). Při řešení úlohy lpi_itest6 je situace obdobná s tím, že volba $K = 10^6$ nezlepší výpočetní vlastnosti NCDK_NR.

Tabulka 4.66: Výsledky pro testovací úlohu lpi_bgprtr při adaptivní a nejlepší volbě parametrů

	Adaptivní volba		Nejlepší volba	
	Počet iterací	Doba výpočtu	Počet iterací	Doba výpočtu
NCDK	123	0.277837	120	0.265369
NCDK_RS	102	0.135802	102	0.130789
NCDK_NR	10 000*	28.962219	10 000*	27.168741
PD	66	0.151366	34	0.084346
PD_RS	66	0.088381	59	0.085107
PD_NR	61	0.085460	61	0.082092
MA	15	0.081933	12	0.064241
MA_RS	12	0.056911	12	0.052338
MA_NR	14	0.056364	14	0.052461
MMA1	25	0.112658	24	0.101547
MMA1_RS	22	0.069154	17	0.060244
MMA1_NR	28	0.074035	27	0.075553
MMA2	73	0.297415	68	0.252885
MMA2_RS	60	0.129217	59	0.133402
MMA2_NR	144	0.268801	96	0.187693
MMA3	22	0.101858	22	0.101127
MMA3_RS	23	0.070365	23	0.070775
MMA3_NR	30	0.076243	26	0.076771
MMA4	93	0.442516	75	0.399489
MMA4_RS	57	0.168929	56	0.170899
MMA4_NR	86	0.115524	42	0.123079
Poznámky: * iterační výpočet ukončen bez rozpoznání neexistence řešení				

Zhodnocení metod při užití adaptivní volby parametrů

Adaptivní volbu parametrů τ a δ lze doporučit pro metodu sledování cesty s dlouhým krokem a základní primárně-duální algoritmus, neboť představuje řešení některých problémů, které při aplikaci těchto metod vznikají. Ani uvedená adaptivní volba ovšem nedokáže vždy rozpoznat neexistenci řešení úlohy. Stejně jako při existenci optimálního řešení nemusí nalézt ε-přesné řešení v nejnižším počtu iterací a za nejkratší dobu. Je třeba také podotknout, že při změně hodnoty parametru γ, ať již se jedná o zvýšení na hodnotu 10^{-2} nebo snížení na 10^{-4} nedochází k selhávání programů, jsou-li uplatněny výše zmíněné úpravy hodnoty parametru K.

Celkové zhodnocení

Spolehlivými metodami pro správnou identifikaci (ne)existence optimálního řešení a pro nalezení ε-přesného řešení, má-li úloha optimální řešení, jsou Mehrotrův algoritmus a jeho modifikace. Modifikace Mehrotrova algoritmu vykazují dobré výsledky i přes provedenou

úpravou, která zapříčinila neplatnost konvergenční teorie. Při použití metod sledování cesty a základního primárně-duálního algoritmu se ukazuje, že není možné používat libovolné kombinace hodnot parametrů a očekávat bezproblémovou funkčnost algoritmů. Řešením problémů vznikajících při „nevhodné“ volbě hodnot parametrů je do jisté míry adaptivní volba parametrů τ a δ.

Závěr

Tato práce shrnula poznatky o metodách vnitřních bodů pro úlohu lineárního programování, seznámila také s metodami sledování cesty (podrobněji s metodou sledování cesty s krátkým krokem a metodou sledování cesty s dlouhým krokem), Mehrotrovým algoritmem a jeho modifikacemi. Předložené programové kódy zpracované v programu MATLAB verze 7.5.0 (R2007b) pro základní primárně-duální algoritmus, "nepřípustnou" variantu metody sledování cesty s dlouhým krokem, Mehrotrův algoritmus a čtyři z jeho modifikací, byly prověřovány na několika úlohách ze sady testovacích úloh lineárního programování LP_NETLIB. Na základě uskutečněných testů bylo provedeno srovnání metod. Dále byla poskytnuta také jistá strategii, jak se vyhnout možné "nevhodné" kombinaci hodnot parametrů způsobující špatné vyhodnocení (ne)existence optimálního řešení nebo nenalezení ε-přesného řešení v „dostatečně krátké“ době. Adaptivní výpočet hodnot parametrů však nelze brát jako univerzální nástroj pro všechní všech problémů, s kterými je možné se při řešení úloh lineárního programování pomocí metod vnitřních bodů setkat.

Literatura

[1] Nocedal, J.,Wright, S. J.: Numerical optimalization, Springer-Verlag, New York, 1999.

[2] Wright, S.: Primal-Dual Interior-Point Methors, SIAM, Philadelphia, 1997.

[3] Dantzig, G. B.: Linear Programming and Extension, Princeton University Press, New Jersey, 1963 (slovenský překlad).

[4] Klee, V., Minty, G. J.: How good is the simplex algorithm? In: Proc. 3rd Symposium Inequalities, Academic Press, New York, 1972, 159-175.

[5] Khachian, L. G.: A polynomial algorithm in linear programming, Soviet Math. Doklady, Vol. 244, No. 1, 1979, 1093-1096 (anglický překlad).

[6] Karmarkar, N.: A new polynomial-time algorithm for linear programming, Combinatorica, Vol. 4, No. 1, 1984, 373-395.

[7] Holeščáková, Z., Písomná práca k dizertačnej skúške - Prediktor - korektor metódy vnútorného bodu riešenia úloh lineárneho programovania, Fakulta matematiky, fyziky a informatiky Univerzity Komenského v Bratislave, Bratislava, 2005 [online] dostupné z: http://pc2.iam.fmph.uniba.sk/institute/halicka/teach/holescakova.pdf, [citováno 1.9.2009].

[8] Ženčák, P.: Texty k přednáškám z lineárního programování - Texty k výuce předmětů LP a LPB, Přírodovědecká fakulta University Palackého v Olomouci, Katedra matematické analýzy a aplikací matematiky, Olomouc, 2007.

[9] Plesník, J., Dupačová, J., Vlach, M.: Lineárne programovanie, Alfa, Bratislava, 1990.

[10] Teorie duality [online] dostupné z: http://www1.osu.cz/studium/mopv2/dualita/priklady/DUALITA.DOC, [citováno 15.9.2009].

[11] Vanderbei, R. J.: Linear Programming: Foundations and Extensions, Kluwer Academic Publishers., Boston, 2001.

[12] Salahi, M., Peng, J., Terlaky, T.: On Mehrotra-Type Predictor-Corrector Algorithms, Technical Report, Advanced Optimization Lab., Department of Computing and Software, McMaster University 2005, [online] dostupné z: http://www.optimization-online.org/DB_HTML/2005/03/1104.html, [citováno 3.10.2009].

[13] Salahi, M., Terlaky, T.: Mehrotra-type predictor-corrector algoritm revised, Optimization Methods and Software, Vol.23, No.2, 2008, 259-273.

[14] Nocedal, J., Wächter, A., Waltz, R. A.: Adaptive Barrier Update Strategies for Nonlinear Interior Methods, 2006.

Seznam obrázků

1.1 Iterační proces znázorněný v prostoru primárních proměnných **x** 13
1.2 Centrální cesta znázorněná v prostoru primárních proměnných **x** 15

2.1 Okolí centrální cesty znázorněné v prostoru primárních proměnných **x** . . . 19

Seznam tabulek

4.1 Složky struktury options 51
4.2 Testovací úlohy LP_NETLIB 52
4.3 Výsledky pro testovací úlohy lp_adlittle a lp_afiro při základní volbě parametrů 53
4.4 Výsledky pro testovací úlohy lp_agg a lp_d2q06c při základní volbě parametrů 54
4.5 Výsledky pro testovací úlohy lp_ship04l a lp_ship04s při základní volbě parametrů 55
4.6 Výsledky pro testovací úlohy lp_ship08l a lp_ship08s při základní volbě parametrů 56
4.7 Výsledky pro testovací úlohy lp_ship12l a lp_ship12s při základní volbě parametrů 57
4.8 Výsledky pro testovací úlohy lpi_bgprtr a lpi_itest6 při základní volbě parametrů 58
4.9 Výsledky pro testovací úlohu lp_adlittle při různých hodnotách parametru γ 60
4.10 Výsledky pro testovací úlohu lp_afiro při různých hodnotách parametru γ . 61
4.11 Výsledky pro testovací úlohu lp_agg při různých hodnotách parametru γ . . 62
4.12 Výsledky pro testovací úlohu lp_d2q06c při různých hodnotách parametru γ 63
4.13 Výsledky pro testovací úlohu lp_ship04l při různých hodnotách parametru γ 64
4.14 Výsledky pro testovací přiklad lp_ship04s pro různé hodnoty parametru γ . 65
4.15 Výsledky pro testovací úlohu lp_ship08l při různých hodnotách parametru γ 66
4.16 Výsledky pro testovací úlohu lp_ship08s při různých hodnotách parametru γ 67
4.17 Výsledky pro testovací přiklad lp_ship12l pro různé hodnoty parametru γ . 68
4.18 Výsledky pro testovací přiklad lp_ship12s pro různé hodnoty parametru γ . 69
4.19 Výsledky pro testovací úlohu lpi_bgprtr při různých hodnotách parametru γ 70
4.20 Výsledky pro testovací úlohy lpi_itest6 při různých hodnotách parametru γ 71
4.21 Výsledky pro testovací úlohy lp_adlittle při různých hodnotách parametru τ 73
4.22 Výsledky pro testovací úlohu lp_afiro při různých hodnotách parametru τ . 74
4.23 Výsledky pro testovací úlohy lp_agg při různých hodnotách parametru τ . . 75
4.24 Výsledky pro testovací úlohu lp_d2q06c při různých hodnotách parametru τ 76
4.25 Výsledky pro testovací úlohu lp_ship04l při různých hodnotách parametru τ 77

4.26 Výsledky pro testovací úlohu lp_ship04s při různých hodnotách parametru τ 78
4.27 Výsledky pro testovací úlohu lp_ship08l při různých hodnotách parametru τ 79
4.28 Výsledky pro testovací úlohu lp_ship08s při různých hodnotách parametru τ 80
4.29 Výsledky pro testovací úlohu lp_ship12l při různých hodnotách parametru τ 81
4.30 Výsledky pro testovací úlohu lp_ship12s při různých hodnotách parametru τ 82
4.31 Výsledky pro testovací úlohu lpi_bgprtr při různých hodnotách parametru τ 83
4.32 Výsledky pro testovací úlohu lpi_itest6 při různých hodnotách parametru τ 84
4.33 Výsledky pro testovací úlohu lp_adlittle při různých hodnotách parametru δ 85
4.34 Výsledky pro testovací úlohu lp_afiro při různých hodnotách parametru δ . 85
4.35 Výsledky pro testovací úlohu lp_agg při různých hodnotách parametru δ . . 86
4.36 Výsledky pro testovací úlohu lp_d2q06c při různých hodnotách parametru δ 86
4.37 Výsledky pro testovací úlohu lp_ship04l při různých hodnotách parametru δ 86
4.38 Výsledky pro testovací úlohu lp_ship04s při různých hodnotách parametru δ 87
4.39 Výsledky pro testovací úlohu lp_ship08l při různých hodnotách parametru δ 87
4.40 Výsledky pro testovací úlohu lp_ship08s při různých hodnotách parametru δ 88
4.41 Výsledky pro testovací úlohu lp_ship12l při různých hodnotách parametru δ 88
4.42 Výsledky pro testovací úlohu lp_ship12s při různých hodnotách parametru δ 88
4.43 Výsledky pro testovací úlohu lpi_bgprtr při různých hodnotách parametru δ 89
4.44 Výsledky pro testovací úlohu lpi_itest6 při různých hodnotách parametru δ 89
4.45 Výsledky pro testovací úlohu lp_adlittle při různých hodnotách parametru β 90
4.46 Výsledky pro testovací úlohu lp_afiro při různých hodnotách parametru β . 91
4.47 Výsledky pro testovací úlohu lp_agg při různých hodnotách parametru β . . 91
4.48 Výsledky pro testovací úlohu lp_d2q06c při různých hodnotách parametru β 92
4.49 Výsledky pro testovací úlohu lpi_bgprtr při různých hodnotách parametru β 92
4.50 Výsledky pro testovací úlohu lpi_itest6 při různých hodnotách parametru β 93
4.51 Výsledky pro testovací úlohu lp_adlittle při nejlepší volbě parametrů 94
4.52 Výsledky pro testovací úlohu lp_afiro při nejlepší volbě parametrů 95
4.53 Výsledky pro testovací úlohu lp_agg při nejlepší volbě parametrů 96
4.54 Výsledky pro testovací úlohu lp_d2q06c při nejlepší volbě parametrů 97
4.55 Výsledky pro testovací úlohu lp_ship04l při nejlepší volbě parametrů 98
4.56 Výsledky pro testovací úlohu lp_ship04s při nejlepší volbě parametrů . . . 99
4.57 Výsledky pro testovací úlohu lp_ship08s při nejlepší volbě parametrů . . . 100
4.58 Výsledky pro testovací úlohu lp_ship12l při nejlepší volbě parametrů 101
4.59 Výsledky pro testovací úlohu lpi_bgprtr při základní volbě parametrů 102
4.60 Výsledky pro testovací úlohu lpi_itest6 při nejlepší volbě parametrů 103
4.61 Výsledky pro testovací úlohu lp_adlittle při adaptivní a nejlepší volbě parametrů 105
4.62 Výsledky pro testovací úlohu lp_afiro při adaptivní a nejlepší volbě parametrů 106

4.63 Výsledky pro testovací úlohu lp_agg při adaptivní a nejlepší volbě parametrů 107
4.64 Výsledky pro testovací úlohu lp_d2q06c při adaptivní a nejlepší volbě parametrů 108
4.65 Výsledky pro testovací úlohu lp_ship08l při adaptivní a nejlepší volbě parametrů 109
4.66 Výsledky pro testovací úlohu lpi_bgprtr při adaptivní a nejlepší volbě parametrů 110