Speciální metody optimalizace

Bc. Bohumil Gajda

Diplomová práce
2009

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2008/2009

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Bohumil GAJDA**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Speciální metody optimalizace**

Zásady pro vypracování:

Práce se zabývá neklasickým vázaným extrémem mnohorozměrové optimalizace ve smyslu kvadratického, konvexního, dynamického programování. Důraz je kladen na ekonomické aspekty využití. Výsledkem budou www materiály, texty, příklady, vzorové příklady a protokoly z uvedené oblasti. Vhodné a kvalifikované prostředí pro simulaci a výpočty Mathematica nebo MATLAB.
Úkoly:

1. **Formulace a řešení úloh konvexního programování.**
2. **Úlohy kvadratického programování.**
3. **Nelineární programování, dynamické programování.**
4. **Vzorové příklady a jejich řešení v prostředí Mathematica.**
5. **Vypracování studijních opor s internetovou aplikací.**

Rozsah práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **Maňas, M.: Optimalizační metody. SNTL Praha 1976.**
2. **Bartko, R.: MATLAB II. Optimalizácia. VŠCHT Praha 2008.**
3. **Šmíd, V.: Konvexní programování. Brno, 1984.**
4. **Pelikán, J.: Diskrétní modely v operačním výzkumu, Professional Publishing, Praha 2001.**
5. **Samek, J., Nečasová, Z., Kodera, J.: Dynamické programování v ekonomických procesech, SPN, Praha 1983. http://www.cis.ohio-state.edu/gurari/course/cis680/cis680Ch21.html.**
6. **Optimization Toolbox User's guide. The MathWorks. Natick, USA 2000.**

Vedoucí diplomové práce: **prof. Ing. Roman Prokop, CSc.**
Ústav automatizace a řídicí techniky

Datum zadání diplomové práce: **20. února 2009**

Termín odevzdání diplomové práce: **27. května 2009**

Ve Zlíně dne 13. února 2009

L.S.

prof. Ing. Vladimír Vašek, CSc.
děkan

doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

Tato práce se zabývá mnohorozměrovými optimalizačními úlohami neklasického vázaného extrému. Je rozdělena do několika částí.

Teoretická část popisuje v první části pojmy a východiska optimalizace. Dále popisuje a kategorizuje jednotlivé typy zpracovávání úloh. V poslední části se zabývá, už podrobně, metodami optimalizace postavenými na neklasickém vázaném extrému. Popisuje podrobně metody konvexního, kvadratického a dynamického programování, včetně postupů, na jednoduchých příkladech.

V praktické části jsou metodou Shetty-Lemke, Whinston-van de Panne a tabulkovou metodou dynamického programování zpracovány úlohy v programu Mathematica. Kód je rozdělen do částí, na nichž je vysvětlen postup zpracování úlohy. Kompletní kódy jsou součástí tištěné i elektronické přílohy.

Klíčová slova: Optimalizace, Klasický vázaný extrém, Neklasický vázaný extrém, Lineární programování, Nelineární programování, Konvexní programování, Kvadratické programování, Dynamické programování.

ABSTRACT

The thesis deals with multidimensional optimization problems of the nonclassical constrained extreme. It is divided into several parts.

In the first part of the theoretical section the terms and the notions of the optimization are defined. In the next part, main categories and methods are analysed and studied. The last part deals in detail with chosen methods of optimization ocused on tasks with inequality constrained problems. The emphasis is laid on the methods of convex, quadratic and dynamic programming, including simple examples and procedures. In the practical section, the special problems are processed in Mathematica environment. There are Shetty-Lemke, Whinston-van de Panne methods and dynamic programming table method. The code is divided into several parts according to algorithms of the method. The complete codes are included in both printed and electronic supplement.

Keywords: Optimization, Classic constrained extrems, Nonclassical constrained extrems, Linear programming, Nonlinear programing, Convex programming, Quadratic programming, Dynamic programming.

Děkuji prof. Ing. Romanu Prokopovi, CSc. nejen za rady a odbornou pomoc, ale také za trpělivost a podporu.

Prohlašuji, že:

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,
že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval.
V případě publikace výsledků budu uveden jako spoluautor.

Ve Zlíně: ....

OBSAH

ÚVOD

Člověk ve svém životě, aniž si to uvědomuje, neustále řeší úlohy optimalizace – optimálního rozhodování. To znamená, že hledá nejvhodnější řešení za daných podmínek. V běžných situacích si pomůže zkušeností. Řešení priorit v rodinném rozpočtu, nákup nejvhodnějšího výrobku, nebo jen rozhodnout co z toho co „nutně potřebuje“ s sebou na dovolenou může zabalit do letadla.

V průběhu 20. století se ovšem optimalizace stala velmi studovanou a atraktivní disciplínou. Pomáhá při hledání nejlepších parametrů u konkrétních výrobků a technologií, ale je také mocným nástrojem pro ekonomy. Může řešit problémy od čistě finančních – nejlepší poměr náklady/výnosy, přes provozní problémy, až po logistiku a marketing.

Z hlediska matematického můžeme optimalizační metody rozdělit do čtyř základních skupin. Zařazení je dáno způsobem, jakým se dosáhne výsledku.

Jednak jsou to *analytické metody*. Ty využívají především výsledků diferenciálního a variačního počtu. Postupy se však ne příliš dobře zpracovávají algoritmicky.

Další jsou *iterační metody*. Ty se postupně v jednotlivých krocích přibližují k řešení – tedy sestavují posloupnosti konvergující k řešení. Pro svou dobrou algoritmizovatelnost se velmi dobře zpracovávají pomocí výpočetní techniky.

Od roku 1944, kdy John von Neumann a Oskar Morgenstern vydali publikaci „Theory of Games and Economic Behavior“ se datuje vznik optimalizační disciplíny *teorie her* (i když základy této metody byly položeny už dříve). Uplatňuje se v mnoha oblastech, včetně těch, které do té doby byly považovány za „nepočítatelné“. Sestavením matematického modelu umožňuje tzv. hledání optimální strategie jednotlivých účastníků.

Speciální metody, označované jako *matematické programování*, nebo *operační analýza* vznikly z neklasického vázaného extrému. Tato práce se zabývá čtvrtou skupinou optimalizace – neklasickým vázaným extrémem mnohorozměrové optimalizace ve smyslu kvadratického, konvexního a dynamického programování. Tyto vznikly jako reakce na požadavky řešení úloh z oblasti ekonomie, vojenství, fyziky... Jejich rozvoj nastal především po druhé světové válce. Dřívější metody řešily především úlohy klasického vázaného extrému, pro něž bylo vyvinuto lineární programování. Část úloh neklasického vázaného extrému pak může být řešena převodem na úlohu lineárního programování.

V teoretické části budou nejdříve objasněny základní pojmy a východiska, na nichž jsou postaveny optimalizační metody jednotlivých typů matematického programování. V samostatné části budou popsány principy nelineárního programování. U metod jmenovaných v zadání práce, to znamená konvexního, kvadratického a dynamického programování, včetně ukázkové úlohy.

Praktická část se bude zabývat rozborem a popisem úloh z teoretické části. Řešení jednotlivých úloh bude popsáno na algoritmu převedeném do prostředí Mathematica.

Protokoly budou členěny dle významných částí právě zpracovávaného kódu s komentáři (jak k části programu, tak k postupu výpočtu) a mezivýsledky. Výsledkem praktické části budou kompletní algoritmy, které budou v celku součástí tištěné přílohy a také budou umístěny na přiloženém CD.

V závěru budou popsané metody zhodnoceny a srovnány z hlediska praktické použitelnosti, složitosti a spolehlivosti.

Součástí elektronické přílohy jsou, kromě kódů pro program Mathematica i studijní opory ve formátu Microsoft PowerPoint a veškeré podklady použité v této práci.

I. TEORETICKÁ ČÁST

1 OPTIMALIZACE

Základní úlohou optimalizace je nalezení optimálního, tedy nejvhodnějšího řešení problému. Z matematického hlediska to znamená nalezení bodu, nebo množiny bodů funkce, které nejlépe vyhovují daným podmínkám. Počet řešení závisí na funkci samotné, jejím definičním oboru a omezeních, za nichž je vyšetřována.

1.1 Základní pojmy

V této části se nejprve dotkneme základních pojmů týkajících se optimalizace obecně. Vzhledem k tomu, že vyšetřování minima a maxima funkce lze matematicky zaměnit: $\min f(x) = \max(-f(x))$, pokud nebudeme hovořit o konkrétním případu, budeme uvádět jen výraz extrém.

1.1.1 Konstrukce optimalizačního modelu

Při řešení optimalizačních úloh sestavujeme model, který se skládá ze dvou částí. První z nich tvoří *omezení* charakterizující základní procesy, které jsou typické a podstatné pro studovanou úlohu.

Jádro problému charakterizují *vlastní omezení*. V praxi jich může být velké množství. Proto je jedním z prvních kroků analýza problému a stanovení podstatných omezení a zanedbání nepodstatných vlivů.

Převážně v ekonomických úlohách se setkáváme s pojmem *podmínky nezápornosti*. Charakterizujeme tím prostý požadavek, aby procesy, které uskutečňujeme, nebyly na záporné úrovni. Ekonomové jsou sice schopni počítat se záporným ziskem (i když i ten můžeme převést na proměnnou „ztráta“ a pohybovat se tak v rámci výpočtů v kladných číslech), ale počet vyrobených kusů je pro ně v záporných číslech nepřijatelný.

Pro některé úlohy, opět převážně z ekonomického sektoru, jsou důležité *podmínky celočíselnosti*. V praxi to znamená, že nás například ve výrobě zajímá jen počet skutečně hotových výrobků za sledované období (směna, stanovený termín...).

Vlastní omezení lze zapsat několika způsoby. Časté je omezení lineárními nerovnicemi.

Pomocí *soustavy nerovnic*:

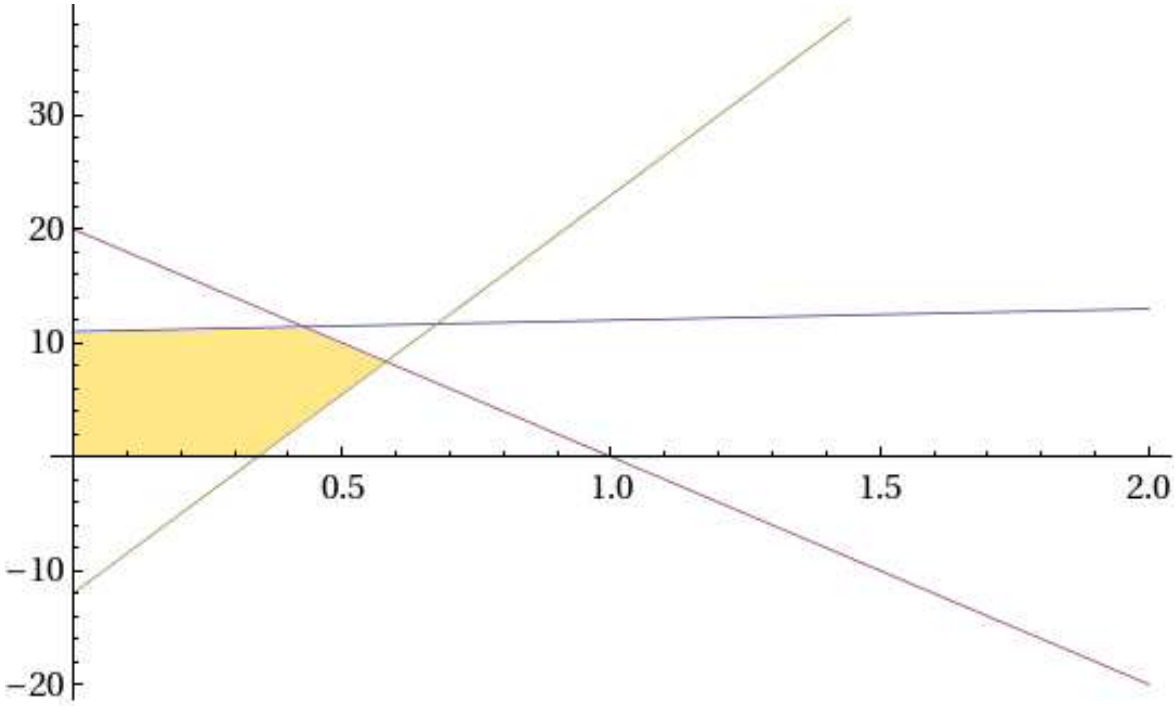
$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &\leqq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &\leqq b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &\leqq b_m \end{aligned} \tag{1}$$

Totéž můžeme zapsat v maticové formě jako:

$$\bar{A}\bar{x} = \bar{b} \quad \text{kde}$$

$$\bar{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \qquad \bar{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix} \tag{2}$$

Při analýze nám může pomoci zobrazení v grafu:



Obr. 1. Znázornění tří lineárních omezení se zvýrazněním průniku – oblasti přípustných řešení (zde tvoří konvexní množinu)

1.1.2 Účelová funkce

Účelovou (extremizovanou) funkcí rozumíme funkci reálných proměnných $f(\bar{x})$ definovanou na množině $\mathrm{M} \in \mathbb{R}^n$, jejíž extrém hledáme. Množina M charakterizuje oblast, pro kterou jsou nalezená řešení korektní, fyzikálně realizovatelná a pod. Může být samozřejmě definována na celé oblasti reálných čísel.

1.1.3 Klasifikace extrémů funkce

Extrém každé funkce je zajištěn *Weierstrassovou větou:*

Věta 1 *Každá spojitá funkce $f(\bar{x})$ definovaná na kompaktní (ohraničené a uzavřené) množině $\mathbf{X} \subset E^n$, má na ní maximální i minimální hodnotu.*

Extrémem funkce rozumíme bod $\bar{x}$, ve kterém funkce nabývá maxima, či minima. V této souvislosti rozlišujeme:

Ostrý vs. neostrý extrém

Funkce f má v bodě $\bar{x}$ *(neostré) lokální maximum*, jestliže existuje okolí bodu $\bar{x}$ takové, že $f(x) \leqq f(\bar{x})$ platí pro všechna x z tohoto okolí [1].

Funkce f má v bodě $\bar{x}$ *ostré lokální maximum*, jestliže existuje okolí bodu $\bar{x}$ takové, že $f(x) < f(\bar{x})$ platí pro všechna x z tohoto okolí, s výjimkou bodu $x = \bar{x}$ [1].

Lokální vs. globální extrém

Je-li X neprázdná množina z euklidovského n-rozměrného prostoru E^n, řekneme, že funkce f má v bodě $\bar{x} \in X$ *lokální maximum vzhledem k* X, je-li f na množině X definována a jestliže existuje okolí bodu $\bar{x}$ takové, že $f(x) \leqq f(\bar{x})$ platí pro všechna x z tohoto okolí, která jsou současně body X [1].

Pokud na stejné množině platí $f(x) < f(\bar{x})$, přičemž $x \neq \bar{x}$, hovoříme o *ostrém globálním maximu vzhledem k* X.

Pokud u předchozích vztahů obrátíme smysl znaménka nerovnosti, hovoříme o *minimu funkce.*

Volný vs. vázaný extrém

Dále dělíme extrém funkce podle množiny, nad níž je vyšetřována. Vyšetřujeme-li extrémy funkce n proměnných na celém prostoru E^n mluvíme o *volných extrémech.* Zajímají-li nás pouze takové body maxima nebo minima, které splňují nějaké další podmínky, mluvíme o *vázaných extrémech* [1]. Zde platí některé výjimky – pokud funkce není definována na celém prostoru E^n, vyšetřujeme ji jen na množině $X \subset E^n$, kde je definovaná, ale hovoříme o volném extrému [1].

Klasický vázaný extrém vs. neklasický vázaný extrém

U vázaného extrému dále rozlišujeme *klasický vázaný extrém*, což znamená omezení typu rovnost: $g_j(x_1, x_2, \ldots, x_n) = 0, \qquad j = 1, 2 \ldots, m \qquad m < n,$

a dále *neklasický vázaný extrém* s omezením typu nerovnost, např.

$g_j(x_1, x_2, \ldots, x_n) \geqq 0, \qquad j = 1, 2, \ldots, m \qquad x_i \leqq 0, i = 1, 2, \ldots, n$

Úlohy, kde účelová (extremalizovaná) funkce splňuje tyto podmínky, nazýváme úlohy *matematického programování.*

Dříve, především před druhou světovou válkou, se vyšetřovaly pouze úlohy klasického vázaného extrému, tedy za podmínek, popsaných výše. V době po druhé světové válce

se objevily nové důležité úlohy z oblasti vojenství, fyziky a ekonomie. Díky potřebě jejich řešení se rozvinulo řešení úloh neklasického vázaného extrému.

1.1.4 Podmínky optimality

Slouží k redukci množiny přípustných řešení a platí pro optimální řešení.

Nutné podmínky, (nejčastěji používané) jsou takové které musí splňovat každé optimální řešení dané úlohy.

Postačující podmínky stanoví, že pokud je splňuje přípustný bod řešení, je automaticky považován za optimální řešení.

Obecně se dá hledání optima popsat jako hledání extrému (extrémů) funkce. Tedy řešení, pro nějž nabývá maximálních (minimálních) hodnot.

1.1.5 Vlastnosti funkce

Pro určování extrémů jsou u funkce důležité některé další informace. Vyšetření průběhu je důležité především pro analytické metody.

Konvexní vs. konkávní funkce

Mějme spojitou funkci $f(x)$ definovanou na intervalu M. Pokud pro každé tři body $a < b < c \in$ M leží každý bod $[c; f(c)]$ nad, nebo na úsečce , která spojuje body $[a; f(a)]$ a $[b; f(b)]$, je funkce na tomto intervalu *konkávní*.

Naopak pokud všechny body leží pod nebo na úsečce spojující oba body, je funkce na tomto intervalu konvexní [12].

Dále rozlišujeme, zda je funkce *ryze konvexní*, či *ryze konkávní*. To slouží k odlišení částí funkce, které jsou lineární. Lineární funkce jsou současně konvexní i konkávní.

Konvexnost a konkávnost funkce určujeme pomocí analytických metod.

Inflexní bod

U jednorozměrných funkcí je dalším zajímavým místem inflexní bod. Je to takový bod, kde funkce přechází z konvexní do konkávní, nebo naopak. K jeho určení používáme nejčastěji analytické metody.

Stacionární bod

Je důležitý pro vyšetřování průběhu funkce a jeho extrémů. Máme-li spojitě diferencovatelnou funkci $f(\bar{x})$ je stacionární bod ten, ve kterém jsou všechny první parciální derivace nulové. V tomto bodě může být extrém, nebo sedlový bod funkce.

Sedlový bod

U vyšetřování vícerozměrných funkcí nehovoříme o inflexi. Sedlový bod je takový stacionární bod, ve kterém má funkce vůči některým proměnným minimum a vůči některým maximum. Svůj název dostal díky vzhledu grafu funkce v euklidovském prostoru E^3. Sedlový bod tedy definuje místo, o kterém jsme schopni na základě analýzy říct, že je z matematického hlediska „zajímavý“. Své využití má například při hledání optimální strategie u teorie her.

Další podrobnosti o vyšetřování průběhu funkce jsou popsány v kapitole (1.2.1).

1.1.6 Konvexní množina

Množina $\chi \subset \mathbb{R}^n$ se nazývá *konvexní*, jestliže s každými dvěma body obsahuje také všechny jejich *konvexní lineární kombinace* tj. pro každé $\bar{x}, \bar{y} \in \chi$ a $\lambda \in (0, 1)$ je také $\lambda\bar{x} + (1 - \lambda)\bar{y} \in \chi$.

Této definici vyhovuje také prázdná množina.

Geometricky lze formulovat definici tak, že množina je konvexní, pokud segment, který spojuje libovolné dva její body je také celý součástí množiny.

Některé množinové operace zachovávají konvexnost množin, např.:

Věta 2 *Průnikem libovolného množství konvexních množin je opět konvexní množina.*

Při hledání extrémů funkce $f(\bar{x})$ je velmi vhodné zjistit, zda výsledná množina (daná omezeními) na níž extrém hledáme splňuje podmínky konvexnosti. Můžeme potom využít vlastnosti:

Věta 3 *Protože $f(x)$ je konvexní funkce a M konvexní množina, je každé lokální minimum zároveň minimem globálním [9].*

1.2 Klasifikace úloh optimalizace

1.2.1 Analytické metody

Analytické metody využíváme především při hledání volného extrému. Jsou založeny na výpočtech derivací.

U *jednorozměrných případů* používáme první a druhou derivaci vyšetřované funkce. Položíme-li první derivaci funkce rovnu nule, tedy $f'(x_0) = 0$ čímž zjistíme místo příslušného extrému. Dalším vyšetřením druhé derivace $f''(x_0)$ zjistíme, zda je v tomto bodě minimum, nebo maximum (pro $f''(x_0) < 0$ jde o maximum – funkce je na vyšetřovaném intervalu konkávní, pro $f''(x_0) > 0$ jde o minimum – funkce je na vyšetřovaném intervalu konvexní, v případě kdy $f''(x_0) = 0$ jde o inflexní bod).

V případě, že i pro druhou derivaci platí $f''(x_0) = 0$, musíme pokračovat v dalším derivování. V takovém případě použijeme definici:

Nechť existuje přirozené číslo n tak, že $f^{(n)}(x_0) \neq 0$ (n-tá derivace) a pro všechna $k < n$ platí, že $f^{(k)}(x_0) = 0$. Potom platí:

a) n je sudé a $f^{(n)} < 0$ => ostré lokální maximum

b) n je sudé a $f^{(n)} > 0$ => ostré lokální minimum

c) n je liché => v tomto bodě má funkce inflexní bod

U *vícerozměrných případů* jde o zevšeobecnění jednorozměrného případu. Místo derivací používáme pojmy gradient a Hessova matice.

Gradient funkce definujeme jako vektor parciálních derivací

$$\mathrm{grad} f(x_1, x_2, \ldots, x_n) = \nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \cdots, \frac{\partial f}{\partial x_n} \right) \tag{3}$$

Hessovu matici potom definujeme pomocí druhých parciálních derivací:

$$H = \nabla^2 f(x_1, x_2, \ldots, x_n) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix} \tag{4}$$

Pro další vyšetření extrémů musíme určit, zda je Hessova matice pozitivně definitní, či negativně definitní. To je údaj odpovídající u jednorozměrného případu tomu, zda je druhá derivace kladná, či záporná. Znamená to spočítat hlavní subdeterminanty matice a definitnost potom určit podle *Sylvestrova kritéria*. Podle něj je matice pozitivně definitní (funkce má v bodě minimum), když jsou všechny hlavní subdeterminanty kladné. Matice je negativně definitní (funkce má v bodě maximum), když se u hlavních subdeterminantů pravidelně střídají znaménka, ale první subdeterminant je záporný.

Pokud má matice v jedné rovině maximum a v druhé rovině minimum, je matice indefinitní (funkce zde má sedlový bod). V ostatních případech je matice tzv. semidefinitní (pozitivně, či negativně), což znamená, že o ní nic nevíme.

U vázaného extrému můžeme použít k vyšetření extrému další metody:

Substituční metoda

Za prvé pokud lze z funkce $g(x_1, x_2) = 0$ vyjádřit vyjádřit některou proměnnou, např. $x_2 = u(x_1)$, kterou můžeme dosadit do funkce $z = f(x_1, x_2) = f(x_1, u(x_1))$, čímž převedeme problém na funkci jedné proměnné. Dále optimalizujeme vhodnou numerickou metodou. Na tuto metodu ale příliš úloh nevede, proto se s ní v praxi nesetkáváme.

Metoda Lagrangeových multiplikátorů (neurčitých koeficientů)

Další analytickou metodou, která je velmi univerzální je *Lagrangeova metoda neurčitých koeficientů*. Vychází z toho, že pokud mají funkce $f(\bar{x})$, $g(\bar{x})$ v okolí bodů křivky $g(\bar{x}) = 0$ totální diferenciál a v každém bodě křivky $g(\bar{x}) = 0$ je alespoň jedna z derivací $\frac{\partial g}{\partial x_1}, \frac{\partial g}{\partial x_2} \cdots$ nenulová, pak aby měla funkce $z = f(\bar{x})$ v bodě $[x_1; x_2]$ křivky $g(\bar{x}) = 0$ lokální extrém, musí existovat taková konstanta λ, že funkce

$$F(x_1, x_2, \ldots, x_n, \lambda_1, \lambda_2, \ldots, \lambda_m) = f(x_1, x_2, \ldots, x_n) + \sum_{j=1}^{m} \lambda_j g_j(x_1, x_2, \ldots, x_n) \tag{5}$$

splňuje podmínky:

$$\frac{\partial F}{\partial x_i}(x_1, x_2, \ldots, x_n, \lambda_1, \lambda_2, \ldots, \lambda_m) = 0 \tag{6}$$

$$\frac{\partial F}{\partial \lambda_j}(x_1, x_2, \ldots, x_n, \lambda_1, \lambda_2, \ldots, \lambda_m) = 0 \tag{7}$$

$$m < n \tag{8}$$

Neznámá konstanta λ se nazývá *Lagrangeův multiplikátor*. Funkce F bývá označována jako *Lagrangeova funkce*.

Hledání vázaných extrémů tedy spočívá v sestrojení funkce F a řešení uvedené soustavy rovnic pro neznámé x_0, y_0, λ. Bod $[x_0; y_0]$ bude extrémem tehdy, je-li druhý diferenciál $\partial^2 F$ v tomto bodě nenulový. V opačném případě nemusí jít o extrém [11].

Tato metoda je základem i pro další specializované optimalizační metody.

1.2.2 Iterační metody

Další možností hledání optima jsou iterační metody. Jejich velkou výhodou je jejich dobrá algoritmizovatelnost, což je předurčuje ke zpracování pomocí výpočetní techniky. Principem je postupné přibližování k hledanému extrému. Toto přibližování znamená, že nemusíme vždy najít přesný bod optima, ale ukončení algoritmu (hledání) může být dáno buď počtem kroků, nebo dosaženou požadovanou přesností. Vzhledem k tomu, že některé metody konvertují pouze k lokálnímu extrému, provádíme většinou několik pokusů s různými výchozími body a ten nejúspěšnější pokus prohlásíme za hledané optimum.

Iterační metody můžeme dělit na tři základní množiny:

1. Komparativní (porovnávací), u kterých nepotřebujeme znát derivace
   - Jednorozměrové
     - Fibonacciho metoda a jí podobná metoda zlatého řezu
     - Powelova metoda

- Vícerozměrové
  - Mapování kriteriálních ploch
  - Box-Wilsonova metoda
  - Metody pravidelného a flexibilního simplexu

2. Gradientní
   - Jednorozměrové
     - Regula Falsi
     - Newtonova metoda
   - Vícerozměrové bez omezení
     - S krátkým krokem
     - S dlouhým krokem
     - Kvazinewtonské metody (DFP – Davidon + Fletcher + Powell, CONGRA – metoda konjugovaných gradientů, PARTAN – metoda paralelních tečen (parallel tangent)
3. Vícerozměrové s omezením typu $\leqq$
   - Metoda projekce gradientu
4. Metody náhodného vyhledávání
   - Jednoduché
   - Adaptivní

1.2.3 Teorie her

Základem je řešení situací s nutností výběru vyjádřených matematickým modelem. Rozlišuje postupy a řešení podle podmínek daného problému. V první řadě je to *počet hráčů* $(1, 2, \ldots, n)$, z toho plyne, zda rozhodování bude konfliktní, či nikoli. U konfliktního rozhodování potom záleží na inteligenci hráčů. Inteligencí je myšleno, zda se hráč snaží vybrat nejlepší možné řešení, nebo vybírá čistě náhodně.

Pokud je ve hře více inteligentních hráčů, rozlišujeme dále řešení *antagonistického* – tedy „soupeřivého“ a *neantagonistického* konfliktu.

Dále u neantagonistického konfliktu rozlišujeme, zda jde o *kooperativní* nebo *nekooperativní* způsob hry, tedy jestli spolu hráči spolupracují. Jde o to, že hráči před rozhodnutím uzavřou smlouvu – zkombinují svoje strategie tak, aby výsledná byla co nejvýhodnější pro oba.

Poslední informací ve hře, je výhra. Ta může být buď přenosná, tedy použitelná v dalším kole, nebo je nepřenosná, tedy na další vývoj hry nemá žádný význam.

Matematický model – základní popis problému je potom definován množinou:

$$\{Q, X_1, X_2, \ldots, X_n, J_1(x_1, x_2, \ldots, x_n), J_2(x_1, x_2, \ldots, x_n), J_n(x_1, x_2, \ldots, x_n)\}$$

Q je množina hráčů, množiny X definují strategie jednotlivých hráčů a J jsou jejich jednotlivé výhry.

Volba strategií je potom dána úhlem pohledu hráče. Rozlišujeme *deskriptivní* hledisko, tedy jak by se v dané situaci rozhodl průměrný jedinec a *normativní* hledisko – v dané situaci objektivně nejlepší.

Z matematického hlediska je u hledání optimální strategie v maticových hrách důležitý sedlový bod. Ten je zde definován jako kombinace strategií hráčů, při jejíž změně si žádný z hráčů nemůže polepšit.

Jiná strategie je však ideální pro hráče „s nadhledem“ a jiná pro přímého účastníka. Jako příklad můžeme uvést střetnutí dvou nepřátelských bojových letounů. Pro hráče – pilota je rozhodně výhodnější strategie nepoškodit ani jedno letadlo, než pro hráče řídícího bitvu ze země (např. zničení obou letounů).

2 ÚLOHY MATEMATICKÉHO PROGRAMOVÁNÍ

Základní úlohou matematického programování je, jako u ostatních optimalizačních úloh, nalezení extrému funkce $f(\bar{x})$, kde hledaný vektor $\bar{x}$ je prvkem množiny X. Množina X je nejčastěji určena nějakými algebraickými vztahy mezi složkami vektoru $\bar{x}$, obecně můžeme říct, že jde o úlohu s omezením typu $\neq$.

Základní úloha metamatického programování je tedy

$$\min \{f(\bar{x}) \quad : \bar{x} \in \mathrm{X} \subset E^n\} \tag{9}$$

kde E^n je n-rozměrný Eukleidův prostor [5].

Většina používaných algoritmů konverguje k lokálním extrémům. Globální extrémy jsou nalezeny pouze v určitých případech – např. konvexnost řešeného problému (Věta (3)).

2.1 Volný extrém – metoda přípustných směrů

Vpřípadě, že omezující množinou je $\mathrm{X} = E^n$, jde o úlohu volného extrému. Řešíme ji pomocí analytické metody *přípustných směrů*.

Vycházíme z toho, že je-li $\bar{x}^*$ bodem lokálního minima, nemůže existovat další přípustný směr (k dalšímu minimu), tedy:

Podmínka prvního řádu:

Věta 4 *Je-li $\bar{x}^*$ bodem lokálního minima funkce $f(\bar{x})$, která má spojité první parciální derivace na množině* X*, pak pro libovolný vektor $\vec{s}$, který je přípustným směrem v bodě $\bar{x}^*$ platí:*

$$\mathrm{grad} f(\bar{x}^*) = \nabla f(\bar{x}^*)\vec{s} \geqq 0 \tag{10}$$

Podmínka druhého řádu:

Věta 5 *Je-li $\bar{x}^*$ bodem lokálního minima funkce $f(\bar{x})$, která má spojité první i druhé parciální derivace na množině* X*, pak pro libovolný vektor $\vec{s}$, který je přípustným směrem v bodě $\bar{x}^*$ platí:*

$$\mathrm{grad} f(x) = \nabla f(\bar{x}^*)\vec{s} \geqq 0 \tag{11}$$

$$\textit{Je-li} \qquad \nabla f(\bar{x}^*)\vec{s} = 0, \qquad \textit{pak} \qquad \vec{s}^T \nabla^2 f(\bar{x}^*) = 0 \tag{12}$$

2.2 Vázaný extrém

2.2.1 Omezení typu rovnost

Řeší optimalizační úlohu:

$$\min \left\{\bar{c}^T \bar{x} \quad : \quad \mathrm{A}\bar{x} = \bar{b}, \quad \bar{x} \geqq 0\right\} \tag{13}$$

kde $\bar{x}$ je vektor počtu optimalizované proměnné, $\bar{c}$ je vektor „cen“ a vektor $\bar{b}$ udává požadované nároky.

Tento typ omezení je v řešen lineárním programováním. V kapitole 2.2.3, jsou na příkladu vysvětleny

2.2.2 Omezení typu nerovnost

U omezení typu nerovnost předpokládáme úlohu:

$$\min\{f(\bar{x}) \,:\, g(\bar{x}) \leqq 0\} \tag{14}$$

K analytickému vyšetření extrémů u neklasického vázaného extrému se používá (Karush-)Kuhn-Tuckerova věta:

Věta 6 *Nechť $\bar{x}^*$ je bod relativního minima a $\bar{x}^*$ je regulerní bod omezení. Pak existují $\bar{\lambda} \in E^n$ a $\bar{\mu} \in E^p$ takové, že:*

$$\nabla f(\bar{x}^*) + \nabla \mathbf{h}(\bar{x}^*)\bar{\lambda} + \nabla \mathbf{g}(\bar{x}^*)\bar{\mu} = \mathbf{0} \tag{15}$$

$$\bar{\mu}^T \mathbf{g}(\bar{x}^*) = \mathbf{0} \tag{16}$$

$$\bar{\mu} \geqq \mathbf{0} \tag{17}$$

2.2.3 Lineární programování

Řeší problém vázaného extrému. Vzhledem k tomu, že tento typ extrému není přímo předmětem této práce, bude zde tato metoda jen zmíněna s krátkou ukázkou kvůli vysvětlení některých pojmů

Množina všech přípustných řešení úlohy lineárního programování je konvexní a uzavřená.

Jsou-li všechny funkce vyskytují se v zadání úlohy matematického programování lineární, a dá-li se tedy úloha psát ve tvaru maximalizovat (minimalizovat) funkci $\bar{c}^T\bar{x}$ při omezeních $\bar{b} - \bar{A}\bar{x} \geqq 0; \quad \bar{x} \geqq 0$, jde o úlohu lineárního programování.

Jde tedy o to, že omezení jsou definována pomocí lineárních funkcí, stejně jako účelová funkce.

Lineární úloha nepatří mezi úlohy neklasického vázaného extrému, ale je zde zmíněna kvůli podobnosti některých postupů a výrazů s úlohami neklasického vázaného extrému. Navíc jsou některé úlohy nelineárního programování převoditelné na úlohu lineárního programování.

Příklad 1 *Jednoduchá ukázka úlohy lineárního programování:*
Řešte úlohu: $\max 10x_1 + 30x_2 + x_3$
při omezeních:

$$\begin{array}{rcrcrcl} 3x_1 & + & 2x_2 & & & \leqq & 10 \\ x_1 & + & x_2 & & & \leqq & 10 \\ & & x_2 & + & x_3 & \leqq & 5 \end{array} \qquad x_1, x_2, x_3 \geqq 0$$

Úlohu budeme řešit pomocí simplexové tabulky:
Hlavní část úlohy spočívá v počítání s maticemi. Používáme principy připočtení jednoho řádku, nebo jeho násobku k jinému.
Z maticového zápisu také vychází:

$$\left(\begin{array}{rrrrrrrl} 3 & 2 & 0 & 1 & 0 & 0 & 10 & \textit{zápis prvního omezení } 3x_1 + 2x_2 + 0x_3 \leqq 10 \\ 1 & 1 & 0 & 0 & 1 & 0 & 10 & \textit{zápis druhého omezení } x_1 + x_2 + 0x_3 \leqq 10 \\ 0 & 1 & 1 & 0 & 0 & 1 & 5 & \textit{zápis třetího omezení } 0x_1 + x_2 + x_3 \leqq 10 \\ -10 & -30 & -1 & 0 & 0 & 0 & & \textit{zápis funkce } f(x_1, x_2, x_3) = 10x_1 + 30x_2 + x_3 \end{array}\right)$$

Nejprve vyplníme první oblast tabulky – hodnoty v prvním až třetím řádku x_1 až x_3 bereme z daných omezení, doplněné o x_4 až x_6 diagonální zápis č. 1, čtvrtý řádek vychází z optimalizované funkce (ale zapisujeme záporné hodnoty).

Nyní vybereme sloupec s nejmenší zápornou hodnotou mezi (x_1 – x_3). Dalším krokem je výpočet pomocné proměnné $\beta = \frac{omezeni}{x_i}$, kde i je index proměnné x vybraného sloupce. Řádek s nejnižší hodnotou proměnné β je pro nás výchozí.

x_1	x_2	x_3	x_4	x_5	x_6	omezení	β
3	**2**	0	1	0	0	10	**5**
1	1	0	0	1	0	10	10
0	1	1	0	0	1	5	5
−10	**−30**	−1	0	0	0	0	

Tab. 1. První krok řešení simplexové tabulky s výběrem hodnot pro další krok

Pomocí výpočtů v matici upravíme řádek (vynásobením, nebo dělením celého řádku) tak, aby vybraná hodnota x_i byla jedna a v ostatních řádcích byla hodnota této proměnné nulová (přičtením násobku řádku k jinému), včetně řádku č. 4.

Opakujeme krok vyhledání sloupce s nejmenší zápornou hodnotou a kroky výběru a úprav opakujeme tak dlouho, dokud je ve čtvrtém řádku záporná hodnota. Dopočítaná tabulka č. 1 (vybírané hodnoty jednotlivých kroků jsou zvýrazněny)

x_1	x_2	x_3	x_4	x_5	x_6	omezení	β
3	**2**	0	1	0	0	10	**5**
1	1	0	0	1	0	10	10
0	1	1	0	0	1	5	5
−10	**−30**	−1	0	0	0	0	
1,5	1	0	0,5	0	0	5	$\leftarrow \infty$
−0,5	0	0	−0,5	1	0	5	$\leftarrow \infty$
−1,5	0	**1**	−0,5	0	0	0	**0**
35	0	**−1**	15	0	0	150	
1,5	1	0	0,5	0	0	5	$\leftarrow \infty$
−0,5	0	0	−0,5	1	0	5	$\leftarrow \infty$
−1,5	0	1	−0,5	0	0	0	0
33,5	0	0	14,5	0	1	150	

Tab. 2. Simplexová tabulka pro řešení úlohy lineárního programování

$x_1 = 0$
$x_2 = 5 \qquad \max f(x_1, x_2, x_3) = 10x_1 + 30x_2 + x_3 = 30 \cdot 50 = 150$
$x_3 = 0$

Kontrola omezení:

$3x_1 + 2x_2 \leqq 10$	$x_1 + x_2 \leqq 10$	$x_2 + x_3 \leqq 5$
$0 + 10 \leqq 10$	$0 + 5 \leqq 10$	$0 + 5 \leqq 5$
$10 \leqq 10$	$5 \leqq 10$	$5 \leqq 5$

Nyní si na příkladu vysvětlíme pojmy *stínová cena*, *citlivostní analýza* a *duální úloha*.

2.2.4 Duální úloha

Věta 7 *Má-li primární úloha řešení, pak má řešení i duální úloha se stejnou hodnotou účelové funkce.*

Duální úlohou k $\max f(\bar{x})$ je doplňková úloha $\min f(\bar{x})$

Byla li tedy původní úloha definována jako

$$\max f(\bar{x}) = \bar{c}^T \bar{x} \tag{18}$$

při omezeních $\mathrm{A}\bar{x} \leqq \bar{b}, \qquad \bar{x} \geqq 0$

Duální úloha k ní bude

$$\min g(\bar{x}) = \bar{b}^T \bar{y} \tag{19}$$

při omezeních $\mathrm{A}^T\bar{y} \leqq \bar{c}, \qquad \bar{y} \geqq 0$

Simplexová metoda nám dává výsledky i k duální úloze. V příkladu (1) to znamená, že minimum duální úlohy vyčteme z posledního řádku ve sloupcích x_1 – x_3.

2.2.5 Citlivostní analýza – stínové ceny

Simplexová tabulka nám dává ještě další informaci – o tak zvaných stínových cenách. Ty nám říkají, jak velký vliv na výsledek bude mít změna některého z omezení. U příkladu (1) tyto hodnoty, *Lagrangeovy koeficienty*, najdeme v posledním řádku ve sloupcích x_4 – x_6. Značí, o kolik musíme navýšit který limit, abychom dosáhli navýšení maxima. Zjistíme tak, zda se „vyplatí“, tedy nakolik ovlivní výsledek případné navýšení některého limitu. Pokud limit nebude navýšen o výslednou hodnotu ze simplexové tabulky, může navýšení zůstat nevyčerpáno.

2.2.6 Sedlové vlastnosti Lagrangeovy funkce

Při řešení úlohy nelineárního programování se ukazuje, že Lagrangeova funkce má v optimálních bodech $\bar{x}^*$ a $\bar{\lambda}^*$ sedlový bod [5].

Tedy, že v jednom směru, například x_1, má minimum a v druhém maximum. Máme-li problém

$$\min\{f(\bar{x}) \quad : \mathbf{g}(\bar{x}) \leqq \mathbf{0}\,;\ \bar{x} \leqq \mathbf{0}\}, \tag{20}$$

potom

Věta 8 *Jestliže* $\bar{x}^* \geqq \bar{0}\,, \bar{\lambda}^* \geqq \bar{0}$ *je sedlovým bodem Lagrangeovy funkce*

$$L(\bar{x}, \bar{\lambda}) = f(\bar{x}) + \bar{\lambda}^T \mathbf{g}(\bar{x}), \tag{21}$$

pak $\bar{x}^*$ *je řešením úlohy (20) [5].*

Existuje celá řada algoritmů nelineárního programování, které jsou založeny na sedlových vlastnostech Lagrangeovy funkce.

2.2.7 Dualita úloh nelineárního programování

Nyní probereme problém duality v úlohách nelineárního programování. Máme úlohu:

$$\min\{f(\bar{x}) \quad : \mathbf{g}(\bar{x}) \leqq \mathbf{0}\} \tag{22}$$

Lagrangeova funkce tedy bude $L(\bar{x}, \bar{\lambda}) = f(\bar{x}) + \bar{\lambda}^T \mathbf{g}(\bar{x})$. Nyní sestrojíme dvě funkce:

$$\varphi(\bar{x}) = \max_{\lambda \geqq 0} L(\bar{x}, \bar{\lambda}) \tag{23}$$

$$\psi(\bar{\lambda}) = \min_{x} L(\bar{x}, \bar{\lambda}) \tag{24}$$

kde $\bar{x}$ není nijak omezeno.

Na nich si definujeme duální úlohy:

$$(I.) \quad \min_{x} \varphi(\bar{x}) = \min_{x} \max_{\lambda \geqq 0} L(\bar{x}, \bar{\lambda}) \tag{25}$$

$$(II.) \quad \max_{\lambda \geqq 0} \psi(\bar{\lambda}) = \max_{\lambda \geqq 0} \min_{x} L(\bar{x}, \bar{\lambda}) \tag{26}$$

Kde $I.$ je totožná s původní úlohou (22) a je *primární úlohou* a $II.$ je k ní potom *duální úloha*.

Mezi primární a duální úlohou platí vztah:

$$\text{(Primární úloha)} \quad \min_{x} \max_{\lambda \geqq 0} L(\bar{x}, \bar{\lambda}) \geqq \max_{\lambda \geqq 0} \min_{x} L(\bar{x}, \bar{\lambda}) \quad \text{(Duální úloha)} \tag{27}$$

Dualita úloh matematického programování je velmi důležitá, neboť řešením duální úlohy se k optimu blížíme zdola. Řešením primární úlohy se k optimu blížíme shora.

2.2.8 Vícekriteriální optimalizace

U reálných úloh máme často více kriterií, podle nichž jsme nuceni rozhodovat. Máme několik možností, jak se s takovým zadáním vypořádat. Některé vedou na vektorové, nebo kompromisní programování. Nejčastěji se snažíme převést problém na skalární – s jedním optimalizačním kritériem. K tomu nám slouží následující metody:

- *Volba vah*

  Zde použijeme částečně vlastnosti separability, když jednotlivá kritéria označíme jako $f_1(\bar{x}$ až $f_m(\bar{x})$ a stanovíme váhy $\alpha_1, \alpha_2, \ldots, \alpha_p$. Problém potom řešíme jako jedno kritérium $f(\bar{x}) = \alpha_1 f_1(\bar{x}) + \alpha_2 f_2(\bar{x}) + \cdots + \alpha_p f_p(\bar{x})$.

- *Cílové programování*

  Vychází ze snahy dosáhnout stanoveného cíle ve všech kritériích. To samozřejmě není možné, proto se snaží co nejvíce se přiblížit konkrétní řešení k cíli. Geometrická interpretace je velmi jednoduchá. Označíme-li $f_i^*(\bar{x})$ jako cíl, kterého chceme dosáhnout u i-tého kritéria, cílové programování vede na úlohu

  $$\min\{\lambda \quad : f_i(\bar{x}) - w_i\lambda \leqq f_i^*(\bar{x}),\ i = 1, 2, \ldots, p,\ \bar{x} \in \mathrm{X},\ w_i > 0\} \tag{28}$$

  Výraz λw_i zde vyjadřuje rozdíl mezi stanoveným cílem a skutečnou hodnotou kritéria. Množina X představuje prostor přípustných řešení.

- *Lexikografické uspořádání kritérií*
  Principem této metody je uspořádat kritéria od nejdůležitějšího po nejméně důležité. Z optimálních řešení potom vybíráme takové, které minimalizuje další kritérium v pořadí a tak postupujeme dále. Varianta *s ε-omezením* minimalizuje vybrané preferenční kritérium a ostatní podrobíme omezením. Potom:
  $$\min_{\bar{x}\in \mathrm{X}}\{f_s(\bar{x}) \quad : f_i(\bar{x} \leqq \varepsilon_i, \quad i = 1, 2, \ldots, p, \quad i \neq s\} \tag{29}$$
  Nevýhodou této metody je nutnost volby ε_i a pevná omezení.

- *Minimaxová optimalizace*
  Tato formulace připomíná problémy z teorie her. Řeší v podstatě stejný problém jako pří cílovém programování, kde volíme všechny váhy w_i stejné a cíl je v počátku. Hledáme takové přípustné řešení, v němž je maximální hodnota některého kritéria minimální:
  $$\min_{\bar{x}\in \mathrm{X}} \max_i f_i(\bar{x}) \tag{30}$$

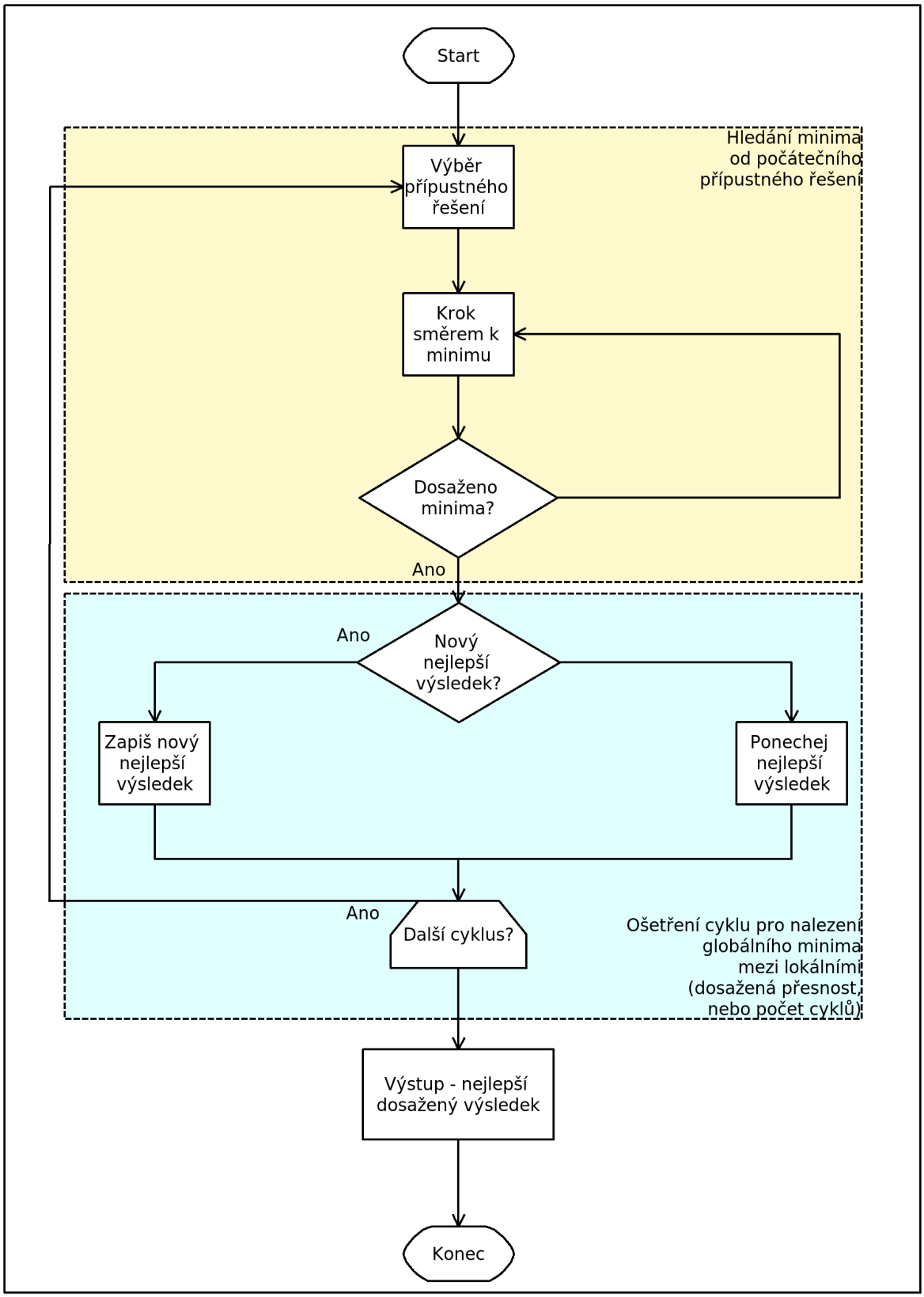
V případě, že nechceme problém převádět na skalární, musíme použít metodu hledání *nedominované varianty*. Problém je dost podobný hledání sedlového bodu u matematického programování (viz kapitolu 1.2.3). Mezi kritérii hledáme takovou rovnováhu, že zlepšení kteréhokoli kritéria je vždy na úkor některého jiného. Pokud je nedominovaných variant více, neznáme exaktní způsob jak určit jedno optimum. Pokud je nedominovaná varianta jen jedna, můžeme ji za optimální označit.

2.3 Nelineární programování

Úlohou nelineárního programování obecně je hledání extrému funkce
$$\min_{x\in M} f(\bar{x}), \tag{31}$$
kde $f(x)$ je obecně nelineární funkce a množina M je popsána soustavou nerovnic
$$g_i(\bar{x}) \leqq 0, \quad i = 1, 2, \ldots, m \quad x_i \geqq 0 \tag{32}$$
kde $g_i(x)$ jsou reálné funkce.

Většina používaných metod patří mezi iterační metody, obecně používáme následující algoritmus:

Sestrojíme nejprve výchozí přípustné řešení (tj. bod z množiny M). Potom se postupně pohybujeme k dalším přípustným bodům, ve kterých je hodnota účelové funkce nižší. Takto pokračujeme dokud je patrná změna účelové funkce, pokud ne, tak výsledný bod prohlásíme za kandidáta na optimum [10]. Vzhledem k tomu, že tyto metody konvertují k lokálním minimům, je třeba proces několikrát opakovat s různými startovacími body (počáteční přípustné řešení). Algoritmus je detailněji popsán na obrázku (2).



Obr. 2. Obecný algoritmus popisující úlohu nelineárního programování

Při řešení úlohy nelineárního programování se ukazuje, že Lagrangeova funkce má v optimálních bodech $\bar{x}^*$ a $\bar{\lambda}^*$ sedlový bod. Pokud tedy máme úlohu:

$$\min\{f(\bar{x}) \quad : g(\bar{x}) \leqq 0\,; \bar{x} \geqq 0\} \tag{33}$$

pak jestliže $\bar{x}^* \geqq 0$, $\bar{\lambda}^* \geqq 0$ je sedlovým bodem Lagrangeovy funkce

$$L(\bar{x}, \bar{\lambda}) = f(\bar{x} + \bar{\lambda}^T g(\bar{x}) \tag{34}$$

je $\bar{x}^*$ je řešením úlohy.

Úlohu nelineárního programování dělíme na dva základní typy:

optimalizace bez vazeb, kdy $M = \mathbb{R}^n$ (tedy při hledání volného extrému). Používáme algoritmy:

- metoda největšího spádu
- metoda sdružených gradientů
- DFP (Davidon-Fletcher-Powell)

optimalizace s vazbami, kdy $M \subset \mathbb{R}^n$ (tedy při hledání vázaného extrému). Používáme algoritmy:

- metody přípustných směrů: Zoutendijk, Frank a Wolf, Topkis-Veinott,...
- Veinottovy opěrné nadroviny
- penalizační algoritmy
- bariérové algoritmy

2.3.1 Klasifikace úloh nelineárního programování

Podle charakteru účelové funkce a omezení můžeme úlohy nelineárního programování rozdělit na:

- *Konvexní programování*
  Účelová funkce i omezení jsou konvexní funkce. K vyšetření používáme Kuhn-Tuckerovy podmínky.
- *Kvadratické programování*
  Jde o druh konvexního programování. Lineární omezení tvoří konvexní množinu a účelová funkce je kvadratická. Lze použít i analytické šetření Kuhn-Tuckerových podmínek, ale existují dva postupy, „Shetty-Lemke“ a „Whinston, van de Panne“, oba postavené na principech práce se simplexovou tabulkou

- *Separabilní programování*
  Účelová funkce je bez smíšených členů:
  $$f(x_1, x_2, \ldots, x_n) = f(x_1) + f(x_2) + \cdots + f(x_n) = \sum_{i=1}^{n} f(x_i) \tag{35}$$
  Toho se využívá například v dynamickém programování, kde se úloha dělí na drobnější, lépe řešitelné problémy

- *Lomené programování*
  Účelová funkce a omezení jsou podíly lineárních funkcí

2.3.2 Konvexní programování

Úlohy pro konvexní programování vychází z předpokladu, že máme množinu X a konvexní funkci reálné proměnné $f : \chi \to \mathbb{R}$ definované na konvexní podmnožině $\chi \in$ X. Optimalizací získáme $\bar{x}^*$ pro které platí: $f(\bar{x}^*) \leqq f(\bar{x});\ \forall\ \bar{x} \in \chi$

Hledání sedlového bodu pomocí metody Lagrangeových multiplikátorů je v úlohách nelineárního programování neefektivní. K řešení je proto používána (Karush)-Kuhn-Tuckerova věta:

Vektor $\bar{x}_0$ je řešením úlohy konvexního programování když a jen když existuje vektor $\bar{\lambda}_0$ takový, že je splněno následujících šest (Kuhn-Tuckerových) podmínek:

$$\bar{v}_0 = \frac{\partial F(\bar{x}_0, \bar{\lambda}_0)}{\partial x_j} \geqq 0 \quad \text{(pro všechna } j\text{)} \tag{36}$$

$$\bar{x}_0^T \bar{v}_0 = \bar{x}_0^T \frac{\partial F(\bar{x}_0, \bar{\lambda}_0)}{\partial x} = \sum_{j=1}^{n} x_{0j} \left(\frac{\partial f(\bar{x}_0)}{\partial x_j} + \sum_{i=1}^{m} l_i \frac{\partial g_i(\bar{x}_0)}{\partial x_j} \right) = 0 \tag{37}$$

$$x_0 \geqq 0 \qquad \text{(nezápornost)} \tag{38}$$

$$\frac{\partial F(\bar{x}_0, \bar{\lambda}_0)}{\partial \lambda_i} = g_i(\bar{x}_0) \leqq 0 \qquad \text{(omezení)} \tag{39}$$

$$\bar{\lambda}_0^T \frac{\partial F(\bar{x}_0, \bar{\lambda}_0)}{\partial \lambda_i} = \sum_{i=1}^{m} \lambda_{0i} g_i(\bar{x}_0) = 0 \tag{40}$$

$$\bar{\lambda}_0 \geqq \bar{0} \tag{41}$$

Tato věta je poměrně složitá. Její použití budeme demonstrovat na jednoduchém příkladu:

Příklad 2 *Hledejte* $\min f(x_1, x_2) = (x_1 - 5)^2 + (x_2 - 5)^2 - 50$. *Množina* M, *na níž je problém definován je daná soustavou nerovnic:*
$x_1 \geqq 1$
$x_1 + x_2 \leqq 4$
$x_1, x_2 \geqq 0$
Upravíme účelovou funkci

$$f(x_1, x_2) = (x_1 - 5)^2 + (x_2 - 5)^2 - 50 = x_1^2 + x_2^2 - 10x_1 - 10x_2$$

Budeme postupovat analytickými metodami. Nejprve zjistíme absolutní minimum (bez omezení) funkce f. *To najdeme v bodě* $[5, 5]$. *Hledaný vázaný extrém, daný našimi omezeními je zřejmě bod, ve kterém se vrstevnice dotýká nejmenší funkční hodnoty z množiny* M. *Je jím zřejmě bod* $[2, 2]$, *jehož platnost ověříme pomocí Kuhn-Tuckerových podmínek:*
$g_1(\bar{x}) \equiv -x_1 \leqq 0$
$g_2(\bar{x}) \equiv x_1 + x_2 - 4 \leqq 0$
Lagrangeova funkce:
$F(\bar{x}, \bar{l}) = f + \lambda_1 g_1 + \lambda_2 g_2 \equiv x_1^2 + x_2^2 - 10x_1 - 10x_2 + \lambda_1(-x_1) + \lambda_2(x_1 + x_2 - 4)$
Její parciální derivace podle x_1 *a* x_2 *jsou:*
$v_1 = 2x_1 - 10 - \lambda_1 + \lambda_2$
$v_2 = 2x_2 - 10 + \lambda_2$
Podle podmínky (40) musí být v optimu $\lambda_i g_i = 0$, *podle toho* $\lambda_1 = 0$. *Podle vztahu (37) je také* $x_i v_i = 0$. *Vzhledem k podmínkám nezápornosti, že* $x_1 > 0$, $x_2 > 0$, *bude i* $v_1 = 0$ *a* $v_2 = 0$. *Z toho plyne, že* $\lambda_2 = 10 - 2x_2 = 10 - 4 = 6$.
Uvedené hodnoty: $x_1 = 2$, $x_2 = 2$
$v_1 = 0$, $v_2 = 0$
$g_1 = -2$, $g_2 = 0$
$\lambda_1 = 0$, $\lambda_2 = 6$
vyhovují Kuhn-Tuckerovým podmínkám (36 – 41).

Jako důkaz použijeme analýzu jednotlivých podmínek:

- Pokud $\bar{x}_0$ leží uvnitř množiny M, při minimalizaci $f(\bar{x}, \bar{\lambda}_0) = \nabla F(\bar{x}, \bar{\lambda}_0) = \bar{0}$

- Leží li $\bar{x}_0$ na hranici, pak podmínky uspořádáme tak, že $g_i(\bar{x}_0)$¡0 pro $i = 1, 2, \ldots, k$ $g_i(\bar{x}_0) = 0$ pro $i = k + 1, k + 2, \ldots, m$. Dále je $\lambda_{0i} = 0$ pro $i = 1, 2, \ldots, k$ a $\dfrac{\partial F(\bar{x}_0, \bar{\lambda}_0)}{\partial x_j} = \dfrac{\partial f}{partial x_j} + \sum\limits_{i=k+1}^{m} \lambda_{0i} \dfrac{\partial g_i(\bar{x}_0, \bar{\lambda}_0)}{\partial x_j} = 0$

- Pokud $\bar{x}_0$ neleží na hranici dané omezeními g_i, ale platí-li pro určité j, že $x_{0j} = 0$ (tedy jde o hranici danou podmínkami nezápornosti), můžeme zavést hraniční funkci $g_{m+1}(\bar{x}) \equiv -x_j \leqq 0$ a rozšířenou Lagrangeovu funkci $F^* = F - \lambda_{m+1} x_j$. Vzhledem k tomu, že $\lambda_1 = \lambda_2 = \cdots = \lambda_m = 0$, je $F = f$ a dále $\dfrac{\partial F^*(\bar{x}_0, \bar{\lambda}_0)}{\partial x_j} = \dfrac{\partial f}{\partial x_j} - \lambda_{0m+1} = 0$, a protože $\lambda_{m+1} = 0$, je $\dfrac{F(\bar{x}_0, \bar{\lambda}_0)}{\partial x_j} = \dfrac{\partial f}{\partial x_j} \geqq 0$.

- Nastanou-li obě předcházející situace současně, bude $\dfrac{\partial F^*}{\partial x_j} = \dfrac{\partial f}{\partial x_j} - \lambda_{0m+1} + \sum\limits_{i=k+1}^{m} \lambda_{0i} \dfrac{\partial g_i(\bar{x}_0, \bar{\lambda}_0)}{\partial x_j} = 0$. První a třetí člen však dává F, takže $\dfrac{\partial F}{\partial x_j} = \lambda_{0m+1} \geqq 0$ [17].

Jak je i z příkladu vidět, analytická metoda není příliš použitelná, navíc je velmi špatně algoritmizovatelná. Proto vznikají pro specifické úlohy konvexního programování specializované postupy, často velmi dobře algoritmizovatelné.

2.3.3 Kvadratické programování

Jedním ze speciálních případů konvexního programování je kvadratické programování. Jak již bylo řečeno v (2.3.1), jde o případ, kdy jsou omezení lineární a účelová funkce je kvadratická.

Úlohu formulujeme jako:

$$J = \min f(\bar{x}) = \min(\bar{c}^T\bar{x} + \frac{1}{2}\bar{x}^T\mathrm{Q}\bar{x}) \qquad \bar{x}, \bar{c} \in \mathbb{R}^n \qquad \mathrm{Q} \in \mathbb{R}^{n\times n} \tag{42}$$

kde $\mathrm{Q} = \mathrm{Q}^T \geqq 0$ je symetrická kladně semidefinitní matice definovaná na konvexní množině definované omezeními lineárními rovnicemi a nerovnicemi:

$$\mathrm{A}\bar{x} \leqq \bar{b} \qquad \bar{b} \in \mathbb{R}^m,\ \mathrm{A} \in \mathbb{R}^{m\times n} \tag{43}$$

při podmínkách nezápornosti:

$$\bar{x} \geqq 0, \qquad \bar{x} \in \mathbb{R}^m \tag{44}$$

Většina používaných metod je založena na řešení Kuhn-Tuckerových podmínek. Ty mají tvar:

$$-\mathrm{Q}\bar{x}^* - \mathrm{A}^T\bar{\lambda}^* + \bar{u}^* = \bar{c} \qquad \bar{u}^* \in \mathbb{R}^n,\ \bar{\lambda}^* \in \mathbb{R}^m \tag{45}$$

$$\mathrm{A}\bar{x}^* + \bar{v}^* = \bar{b} \qquad \bar{v}^* \in \mathbb{R}^n \tag{46}$$

kde $\bar{x} \geqq \bar{0}$, $\bar{u} \geqq \bar{0}$, $\bar{\lambda} \geqq \bar{0}$ a $\bar{v} \geqq \bar{0}$ jsou čtyři nezáporné vektory, které splňují následující podmínky:

$$x_i^* u_i^* = 0 \qquad i = 1, 2, \ldots, n \tag{47}$$

$$\lambda_j^* v_j^* = 0 \qquad i = 1, 2, \ldots, m \tag{48}$$

Teď sestrojíme Lagrangeovu funkci:

$$L(\bar{x}, \bar{\lambda}) = \bar{c}^T\bar{x} + \frac{1}{2}\bar{x}^T\mathrm{Q}\bar{x} + \bar{\lambda}^T(\mathrm{A}\bar{x} - \bar{b}) \tag{49}$$

Z ní dostaneme:

$$\bar{x}^* \geqq 0 \tag{50}$$

$$\bar{\lambda}^* \geqq 0 \tag{51}$$

$$\nabla_{\bar{x}} L = \bar{c} + \mathrm{Q}\bar{x}^* + \mathrm{A}^T\bar{\lambda}^* \stackrel{!}{=} \bar{u}^* \geqq 0 \tag{52}$$

$$-\nabla_{\bar{\lambda}} L = \bar{b} - \mathrm{A}\bar{x}^* \stackrel{!}{=} \bar{v}^* \geqq \bar{0} \tag{53}$$

$$\bar{u}^{*T}\bar{x}^* = \left(\bar{c} - \mathrm{Q}\bar{x}^* + \mathrm{A}^T\bar{\lambda}^*\right)^T\bar{x}^* = 0 \tag{54}$$

$$\bar{v}^{*T}\bar{\lambda}^* = \left(\bar{b} - \mathrm{A}\bar{x}^*\right)^T \bar{\lambda}^* = 0 \tag{55}$$

Zjednodušeně můžeme zapsat:

$$\mathrm{M} = \begin{pmatrix} \mathrm{Q} & \mathrm{A}^T \\ -\mathrm{A} & \bar{0} \end{pmatrix};\ \bar{z} = \begin{pmatrix} \bar{x} \\ \bar{\lambda} \end{pmatrix} \geqq \bar{0};\ \bar{w} = \begin{pmatrix} \bar{u} \\ \bar{v} \end{pmatrix} \geqq \bar{0};\ \bar{q} \begin{pmatrix} \bar{c} \\ \bar{b} \end{pmatrix} \tag{56}$$

a dále:

$$\left(\bar{u}^{*T}\ \bar{v}^{*T}\right) \begin{pmatrix} \bar{x}^* \\ \bar{\lambda}^* \end{pmatrix} = \bar{w}^{*T}\bar{z}^* = 0 \tag{57}$$

čímž jsme řešení systému Kuhn-Tuckerových podmínek zredukovali na hledání takových nezáporných vektorů $\bar{z}^* \geqq \bar{0}$ a $\bar{w}^* \geqq \bar{0}$, které splňují předcházející rovnice. Takto formulovaný problém se nazývá *Lemkeho úloha o komplementárnosti proměnných.* [16]

Metoda Shetty-Lemke

Jedna z metod, jak řešit úlohu kvadratického programování pomocí tabulky. Pro její použití byla autory zavedena pomocná proměnná y do řádků, ve kterých má vektor $\bar{q}$ záporné položky. Této proměnné potom přiřadíme zápornou hodnotu té složky q_s, jež má ze záporných q_i nejvyšší absolutní hodnotu.

$$y = -\min_j(q_j) = -q_s \text{ pro } q_j < 0 \tag{58}$$

Dále označíme symbolem $\bar{l} \in \mathbb{R}^{n+k}$, který bude mít hodnotu $l_i = 1$ pro $q_s < 0$ a $l_i = 0$ pro $q_i \geqq 0$.

Z toho nám vyplyne upravená rovnice:

$$\bar{w} - \mathrm{M}\bar{z} - \bar{l}y = \bar{q} \tag{59}$$

Tabulka kvadratického programování potom bude mít základ:

P.b.	q	w_1	$\ldots$	w_s	$\ldots$	w_{n+k}	z_1	$\ldots$	z_s	$\ldots$	z_{n+k}	y
w_1	q_1	1	$\ldots$	0	$\ldots$	0	$-m_{1,1}$	$\ldots$	$-m_{1,s}$	$\ldots$	$-m_{1,n+k}$	l_1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
w_s	q_s	0	$\ldots$	1	$\ldots$	0	$-m_{s,1}$	$\ldots$	$-m_{s,s}$	$\ldots$	$-m_{s,n+k}$	-1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
w_{n+k}	q_{n+k}	0	$\ldots$	0	$\ldots$	1	$-m_{n+k,1}$	$\ldots$	$-m_{n+k,s}$	$\ldots$	$-m_{n+k,n+k}$	0

Tab. 3. Tabulka pro kvadratické programování metodou Shetty-Lemke

Nyní zařadíme mezi bázické proměnnou y, která nahradí w_s. Potom úpravami známými ze simplexové tabulky v lineárním programování dostaneme (viz následující tabulku).

Poté co vypadla bázická proměnná w_s, nahradila ji, na principu komplementárnosti proměnných z_s. Tím je určen sloupec s klíčovým prvkem. Nyní postupujeme stejně jako u simplexové tabulky lineárního programování.

P.b.	q	w_1	$\dots$	w_s	$\dots$	w_{n+k}	z_1	$\dots$	z_s	$\dots$	z_{n+k}	z_{n+k+1}
w_1	q_1	1	$\dots$		$\dots$	0	$-m_{1,1}$	$\dots$	$-m_{1,s}$	$\dots$	$-m_{1,n+k}$	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
y	q_s	0	$\dots$	-1	$\dots$	0	$-m_{s,1}$	$\dots$	$-m_{s,s}$	$\dots$	$-m_{s,n+k}$	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
w_{n+k}	q_{n+k}	0	$\dots$		$\dots$	1	$-m_{n+k,1}$	$\dots$	$-m_{n+k,s}$	$\dots$	$-m_{n+k,n+k}$	0

Tab. 4. Tabulka pro kvadratické programování metodou Shetty-Lemke po úpravě v prvním kroku

Příklad 3 *Pomocí metody Shetty-Lemke hledáme*
$\min f(x_1, x_2) = (x_1 - 2x_2 + 1)^2 + (x_2 - 2)^2 = x_1^2 + 2x_1 - 4x_j x_2 - 8x_2 + 5x_2^2 + 5$
Za daných omezení:
$x_1 + 2x_2 \leqq 4$
$x_1 + x_2 \leqq 3$
a při splněných podmínkách nezápornosti $x_1 \geqq 0 \quad x_2 \geqq 0$

Můžeme provést maticový zápis omezení:

$$\mathrm{A}\bar{x} = \begin{pmatrix} 1 & 2 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leqq \begin{pmatrix} 4 \\ 3 \end{pmatrix} = \bar{b}$$

Stejným způsobem zapíšeme účelovou funkci:

$$f(\bar{x}) = 5 + \begin{pmatrix} x_1 & x_2 \end{pmatrix} \begin{pmatrix} 2 \\ 8 \end{pmatrix} + \frac{1}{2} \begin{pmatrix} x_1 & x_2 \end{pmatrix} \begin{pmatrix} 2 & -4 \\ -4 & 10 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = a + \bar{c}^T \bar{x} + \frac{1}{2} \bar{x}^T \mathrm{Q} \bar{x}$$

Dále máme vektor

$$\bar{z} = \begin{pmatrix} x_1 & x_2 & \lambda_1 & \lambda_2 \end{pmatrix}$$

a dále množinu M

$$\mathrm{M} = \begin{pmatrix} \mathrm{Q} & \mathrm{A}^T \\ -\mathrm{A} & \bar{0} \end{pmatrix} \begin{pmatrix} 2 & -4 & 1 & 1 \\ -4 & 10 & 2 & 1 \\ -1 & -2 & 0 & 0 \\ -1 & -1 & 0 & 0 \end{pmatrix} \quad \text{a} \quad \bar{q} = \begin{pmatrix} \bar{c} \\ \bar{b} \end{pmatrix} = \begin{pmatrix} 2 & -8 & 4 & 3 \end{pmatrix}^T$$

Podle výše popsaného postupu nyní můžeme doplnit údaje do tabulky kvadratického programování a dopočítat hodnoty.

Řádky a sloupce vybrané pro další krok jsou zvýrazněny žlutým podkladem. Pokračujeme standardními metodami zpracování jako u simplexové tabulky v kombinaci s výběry dle algoritmu pro metodu Shetty-Lemke. Kompletní zpracování je v následující tabulce.

$P.b.$	q	w_1	w_2	w_3	w_4	z_1	z_2	z_3	z_4	y	p
w_1	2	1	0	0	0	-2	4	-1	-1	0	
w_2	-8	0	1	0	0	4	-10	-2	-1	-1	8
w_3	4	0	0	1	0	1	2	0	0	0	
w_4	3	0	0	0	1	1	1	0	0	0	
w_1	2	1	0	0	0	-2	4	-1	-1	0	0,5
y	8	0	-1	0	0	-4	10	2	1	1	0,8
w_3	4	0	0	1	0	1	2	0	0	0	2
w_4	3	0	0	0	1	1	1	0	0	0	3
z_2	0,5	0,25	0	0	0	$-0,5$	1	$-0,25$	$-0,25$	0	
y	3	$-2,5$	-1	0	0	1	0	4,5	3,5	1	3
w_3	3	$-0,5$	0	1	0	2	0	0,5	0,5	0	1,5
w_4	2,5	$-0,25$	0	0	1	1,5	0	0,25	0,25	0	$\frac{5}{3}$
z_2	1,25	0,125	0	0,25	0	0	1	$-0,125$	$-0,125$	0	
y	1,5	$-2,25$	-1	$-0,5$	0	0	0	4,25	3,25	1	$\frac{6}{17}$
z_1	1,5	$-0,25$	0	0,5	0	1	0	0,25	0,25	0	6
w_4	0,25	0,125	0	$-0,75$	1	0	0	$-0,125$	$-0,125$	0	
z_2	$\frac{44}{34}$	$\frac{2}{34}$	$-\frac{1}{34}$	$\frac{8}{34}$	0	0	1	0	$-\frac{1}{34}$	$-\frac{1}{34}$	x_2^*
z_3	$\frac{12}{34}$	$-\frac{18}{34}$	$-\frac{8}{34}$	$-\frac{4}{34}$	0	0	0	1	$\frac{26}{34}$	$\frac{8}{34}$	λ_1^*
z_1	$\frac{48}{34}$	$-\frac{4}{34}$	$\frac{2}{34}$	$\frac{18}{34}$	0	1	0	0	$\frac{2}{34}$	$-\frac{2}{34}$	x_1^*
w_4	$\frac{10}{34}$	$\frac{2}{34}$	$-\frac{1}{34}$	$-\frac{26}{34}$	1	0	0	0	$-\frac{1}{34}$	$\frac{1}{34}$	

Tab. 5. Tabulka kvadratického programování pro výpočet příkladu č. (3) metodou Shetty-Lemke

Výsledné hodnoty jsou zvýrazněny zeleně. Máme tedy:

$$x_1^* = \frac{24}{17}$$

$$x_2^* = \frac{22}{17}$$

$$\lambda_1^* = \frac{6}{17}$$

$$\lambda_2^* = 0$$

Po dosazení do původní funkce $f^*(x_1, x_2) = f(x_1^*, x_2^*) = \frac{9}{17}$.

Kontrolou můžeme potvrdit platnost omezení:

$$x_1 + 2x_2 \leqq 4 \quad \rightarrow \frac{24}{17} + 2\frac{22}{17} \leqq 4 \quad \rightarrow 4 \leqq 4$$

$$x_1 + x_2 \leqq 3 \quad \rightarrow \frac{24}{17} + \frac{22}{17} \leqq 3 \quad \rightarrow 2,71 \leqq 3$$

Metoda Whinston, van de Panne

Vychází z metody Shetty-Lemke. Používá i podobnou metodu zápisu do tabulky, jen vynechává proměnnou y a používá tedy trochu jinou metodu modifikace (hned v prvním kroku přechází mezi bázické proměnná z_s).

Z tabulky je patrno, jakým způsobem jsou použity proměnné z maticového zápisu. Oproti chybějící pomocné proměnné y je navíc použita hodnota $-\mathrm{M}$ (pravá část pod sloupci $\bar{x}$ a $\bar{\lambda}$).

P.b.	$\bar{q}$	$\bar{u}$	$\bar{v}$	$\bar{x}$	$\bar{\lambda}$
$\bar{u}$	$\bar{c}$	$\bar{1}$	$\bar{0}$	$-\mathrm{Q}$	$-\mathrm{A}^T$
$\bar{v}$	$\bar{b}$	$\bar{0}$	$\bar{1}$	A	$\bar{0}$

Tab. 6. Tabulka pro výpočet metodou Whinston, van de Panne

Začíná se bázickým řešením, kdy nebázické proměnné jsou $\bar{x} = 0$ a $\bar{\lambda} = 0$ a bázi tvoří $\bar{v} = \bar{b} \geqq \bar{0}$ a aspoň v jednom případě je porušena podmínka $\bar{u} = \bar{c} \geqq \bar{0}$, jinak by nebylo co řešit.

Klíčový řádek se vybírá podle pravidla:

$$p_t = \frac{q_t}{m_{ts}} = \min_j \left\{ \frac{q_j}{m_{js}} \;\middle|\; \frac{q_j}{m_{js}} > 0 \wedge m_{js} \neq 0 \right\} \tag{60}$$

takže bázickou se stane proměnná s indexem t. Tomu předchází výběr s-tého sloupce pro určení proměnné w_s, která přejde mezi nebázické. Budeme přitom rozlišovat dvě situace:

- standardní, kdy jedna proměnná z dvojice w_i a z_i patří mezi bázické a druhá mezi nebázické proměnné
- nestandardní, kdy w_j i z_j jsou obě bázické, nebo obě nebázické.

Při výběru řídícího sloupce ve standardní situaci, kdy platí princip komplementárnosti, se postupuje tak, že klíčový bude sloupec té proměnné z_k, jehož „dvojče“ w_k má mezi zápornými nejvyšší absolutní hodnotu.

V nestandardním případě to bude sloupec proměnné z_k z toho páru z_k a w_k, který nemá zastoupení v bázi. Pokud by nebylo možné volit z_k, například kvůli hrozbě zacyklení, nebo protože neexistuje $p_s > 0$, zvolíme sloupec komplementární proměnné w_k [16].

V porovnání s metodou Shetty-Lemke je tato metoda jednodušší a spolehlivější.

Příklad 4 *Nyní metodou Whinston, van de Panne spočítáme příklad se stejným zadáním, jako příklad (3). V prvním kroku budou použity stejné hodnoty, jen místo* M *budou dosazeny hodnoty* $-\mathrm{M}$*, tedy* $\begin{pmatrix} -\mathrm{Q} & -\mathrm{A}^T \\ \mathrm{A} & \bar{0} \end{pmatrix}$.

Výběr sloupce hodnoty p dopočítáme podle vztahu (60), v prvním kroku bude vybrán sloupec z_2 *a po dopočítání řádek* w_1.

P.b.	q	w_1	w_2	w_3	w_4	z_1	z_2	z_3	z_4	p
w_1	2	1	0	0	0	-2	4	-1	-1	$0,5$
w_2	-8	0	1	0	0	4	-10	-2	-1	$0,8$
w_3	4	0	0	1	0	1	2	0	0	2
w_4	3	0	0	0	1	1	1	0	0	3
z_2	$0,5$	$0,25$	0	0	0	$-0,5$	1	$-0,25$	$-0,25$	
w_2	-3	$2,5$	1	0	0	-1	0	$-4,5$	$-3,5$	3
w_3	3	$-0,5$	0	1	0	2	0	$0,5$	$0,5$	$\frac{3}{2}$
w_4	$2,5$	$-0,25$	0	0	1	$1,5$	0	$0,25$	$0,25$	$\frac{5}{3}$
z_2	$1,25$	$0,125$	0	$0,25$	0	0	1	$-0,125$	$-0,125$	
w_2	$-1,5$	$2,25$	1	$0,5$	0	0	0	$-4,25$	$-3,25$	$\frac{6}{17}$
z_1	$1,5$	$-0,25$	0	$0,5$	0	1	0	$0,25$	$0,25$	6
w_4	$0,25$	$0,125$	0	$-0,75$	1	0	0	$-0,125$	$-0,125$	
z_2	$\frac{44}{34}$	$\frac{2}{34}$	$-\frac{1}{34}$	$\frac{8}{34}$	0	0	1	0	$-\frac{1}{34}$	x_2^*
z_3	$\frac{12}{34}$	$-\frac{18}{34}$	$-\frac{8}{34}$	$-\frac{4}{34}$	0	0	0	1	$\frac{26}{34}$	λ_1^*
z_1	$\frac{48}{34}$	$-\frac{4}{34}$	$\frac{2}{34}$	$\frac{18}{34}$	0	1	0	0	$\frac{2}{34}$	x_1^*
w_4	$\frac{10}{34}$	$\frac{2}{34}$	$-\frac{1}{34}$	$-\frac{26}{34}$	1	0	0	0	$-\frac{1}{34}$	

Metodou Whinston, van de Panne bylo dosaženo stejných výsledků, jako u předchozí metody:

$$x_1^* = \frac{24}{17}$$

$$x_2^* = \frac{22}{17}$$

$$\lambda_1^* = \frac{6}{17}$$

$$\lambda_2^* = 0$$

Po dosazení do původní funkce $f^*(x_1, x_2) = f(x_1^*, x_2^*) = \frac{9}{17}$.

Ověření výsledků viz příklad (3).

Vzhledem k eliminaci prvního kroku s proměnnou y je výpočet o jeden krok kratší (výpočet jednodušší a bez úvodního kroku i lépe algoritmizovatelný) a jak již bylo řečeno, tato metoda by měla být i spolehlivější.

2.3.4 Dynamické programování

Dynamické programování je velmi účinný nástroj, používaný k řešení optimalizace v mnoha oborech. Metody a principy na nichž je postaveno jsou spojovány s pracemi amerického matematika Richarda Bellmana.

Hlavní myšlenkou dynamického programování je rozklad problému na snadněji řešitelné podproblémy. Jednotlivé mezivýsledky se potom ukládají pro další použití. Tento princip se nazývá *princip invariantního vnoření*

Jednoduchost jejího základního prinicpu – *principu optimality* (Bellmanův princip optimality) ji činí velmi přitažlivou pro zpracování pomocí výpočetní techniky:

Věta 9 *Optimální trajektorie z libovolného bodu T do bodu Z nezáleží na trajektorii, která představuje cestu z bodu A do T [7].*

Výsledkem dynamického programování je vždy nalezení globálního extrému, i když řešení nemusí být jednoznačné (v případě více globálních extrémů).

Nevýhodou této metody je prudký nárust potřebných prostředků s rostoucím počtem instancí. Díky rekurentním výpočtům a uchovávání mezivýsledků je to především paměť.

Způsoby řešení patří do množiny takzvaného separovatelného programování, využívají principu

$$f(x_1, x_2 \ldots, x_n) = \sum_{i=1}^{n} f(x_i) \tag{61}$$

Jako příklad můžeme uvést: $f(x_1, x_2, x_3) = 0,2x_1^2 + 0,6x_2^3 + 0,1x_3^3$

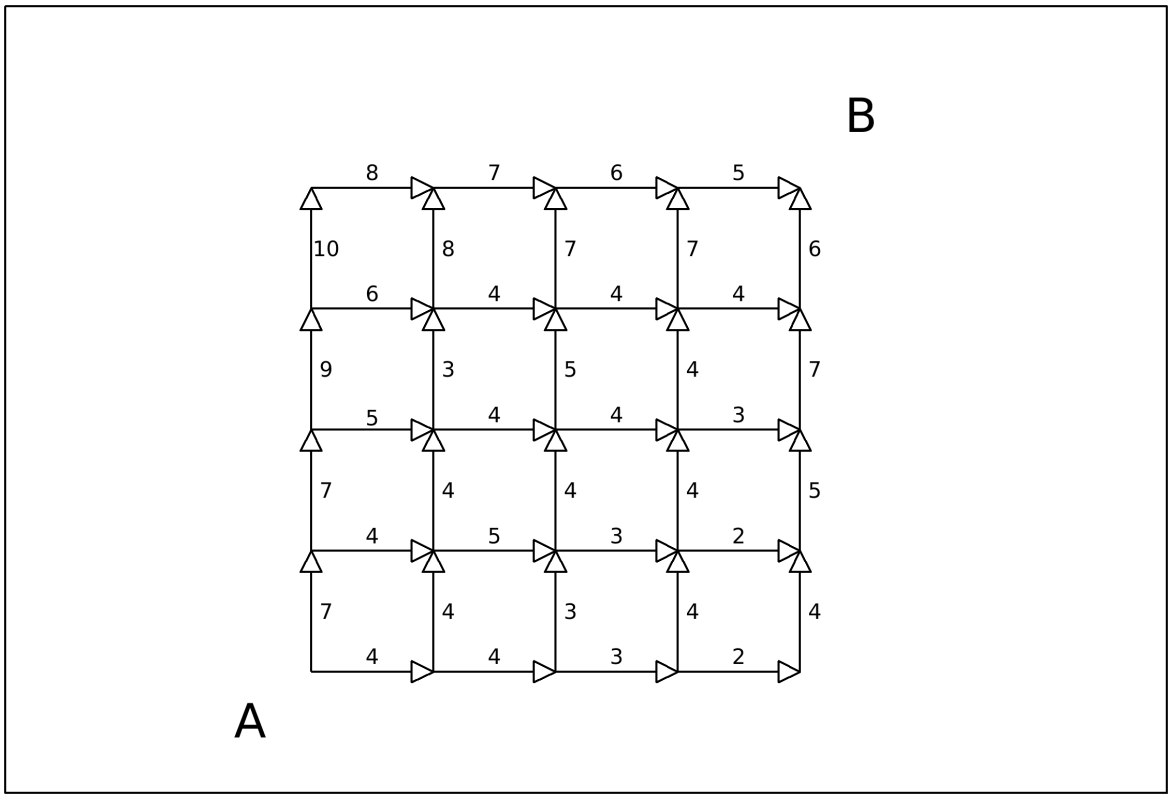
Jedním ze způsobů řešení úlohy dynamického programování je *síťová forma* – tzv. dopravní úloha. Jde o převod problému do orientovaného síťového grafu a nalezení optimální trajektorie jejího průchodu.

Graf je zde reprezentován jednotlivými body (u typické dopravní úlohy jsou to například zastávky autobusu) a orientované hrany představují například náklady (vzdálenost, spotřebu...) na pohyb z jednoho bodu do druhého.

Principem je dvoukolový průchod. Začínáme v cílovém bodě grafu a postupně couváme a ohodnocujeme jednotlivé hrany. Poté co se dostaneme do startovacího bodu, projdeme opět grafem, už po optimální nalezené cestě.

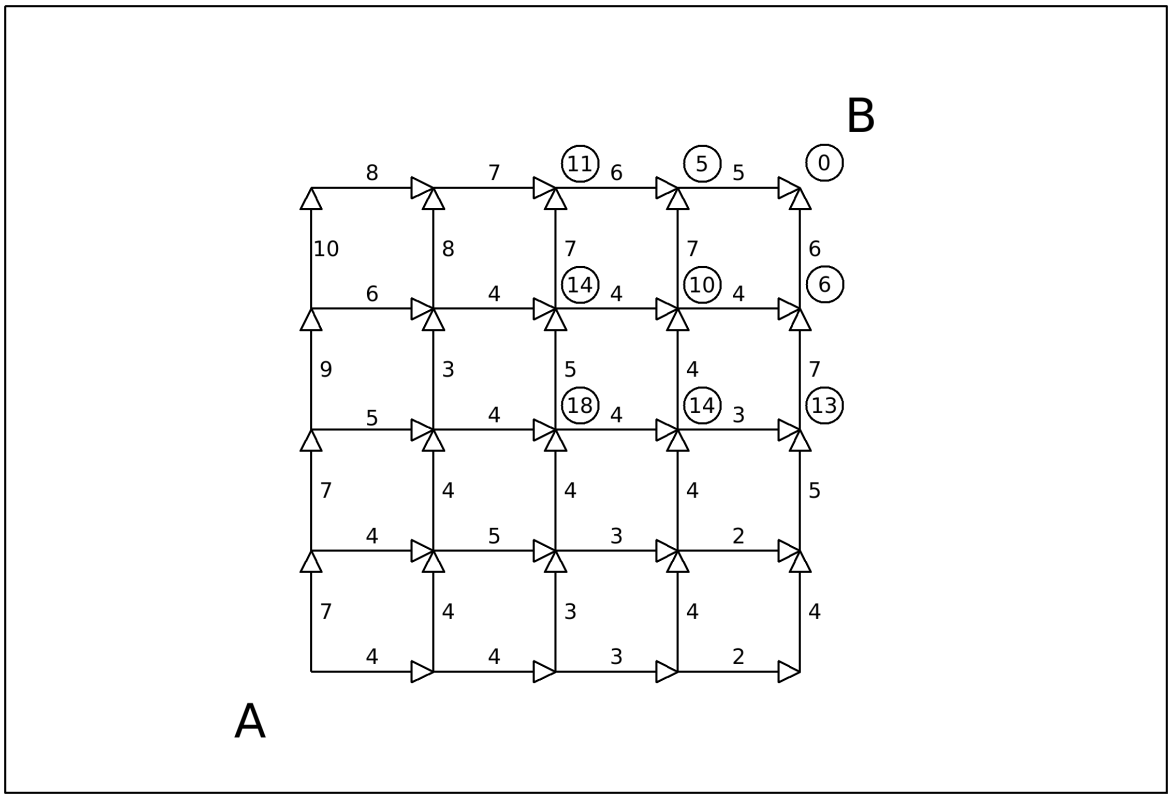
Příklad 5 *Máme orientovaný graf s počátkem v bodě* A *a koncovým bodem* B *– viz obrázek 3 Jednotlivé body znázorňující dopravní uzly. Hodnota hrany c vyjadřuje cenu za přepravu mezi body.*
Hledáme nejlevnější cestu, tedy

$$\min \sum_{i,j} c_{i,j}$$

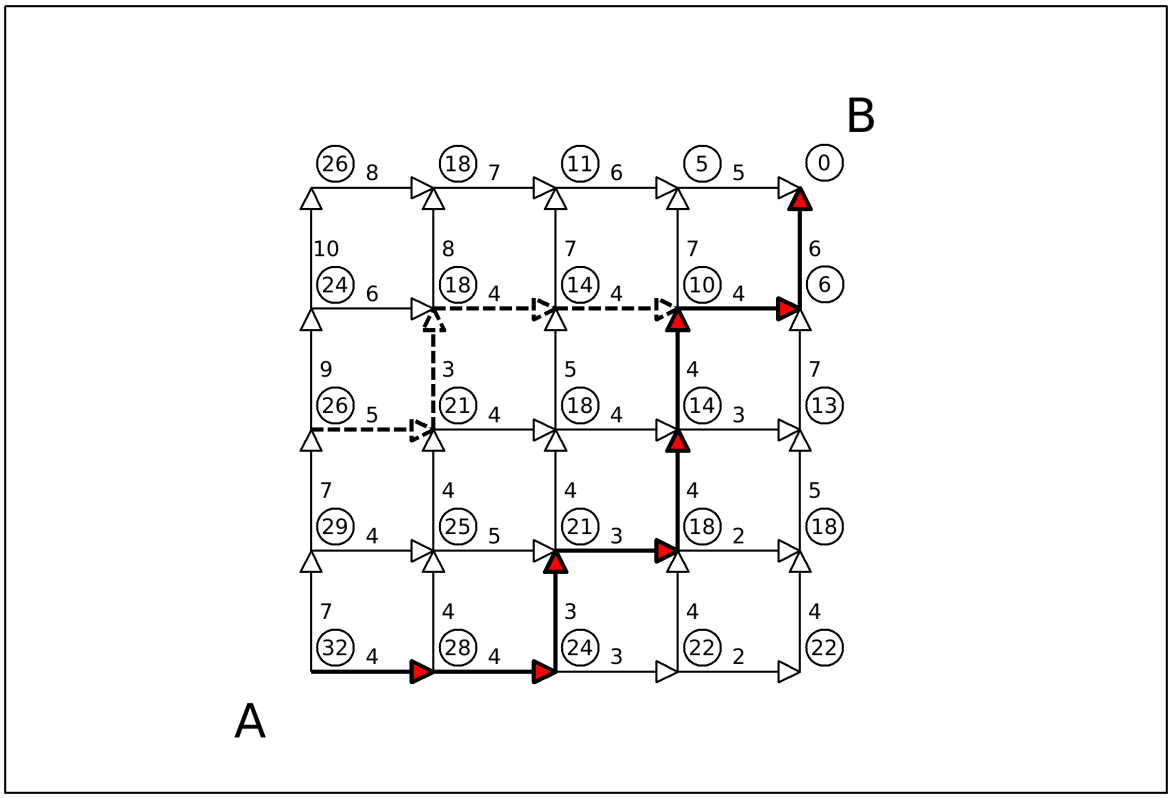


Obr. 3. Znázornění problému řešeného dynamickým programováním v orientovaném síťovém grafu

Postupujeme od bodu B a u průběžných bodů zaznamenáváme optimální hodnoty, tedy načítáme nejoptimálnější cestu z bodu B k tomuto bodu (v kroužku – viz obrázek 4).
Postupujeme, dokud tímto způsobem nemáme ošetřenu cestu až k bodu A.
Ve druhém kole postupujeme od výchozího bodu A po už předpočítané cestě – obrázek 5.
Na obrázku je naznačena (šrafovaně) i cesta, která se do určitého momentu „tvářila“ jako optimální. Vzhledem k tomu, že platí pravidlo, že optimální řešení existuje, musí být nalezena i optimální cesta.
Dále platí, že řešení nemusí být jednoznačené, takže optimálních cest můžeme nalézt několik.



Obr. 4. Znázornění zpětného průchodu síťovým grafem



Obr. 5. Znázornění druhého (dopředného) průchodu síťovým grafem

Dalším způsobem řešení je *tabulková forma* – využívající separovatelnou účelovou funkci.

Příklad 6 *Řešte úlohu dynamického programování:*

$$\min f(x_1, x_2, x_3) = x_1^2 + \sqrt{x_2} + x_3^2 sin\left(x_3 \frac{\pi}{2}\right)$$

při omezeních:

$$x_1 + x_2 + x_3 = 1 \qquad \wedge \qquad x_1, x_2, x_3 \geqq 0; \qquad s\ krokem\, h = 0,1$$

Řešení bude probíhat ve třech krocích:

- *V prvním kole stanovíme hodnoty funkce pro jednotlivá x_1. Velikost proměnné je určena jednak limitem $x_1 + x_2 + x_3 = 1$, krokem h a podmínkou nezápornosti $x_1, x_2, x_3 \geqq 0$. V tomto kole tedy předpokládáme, že ostatní proměnné jsou nulové a počítáme jen separovanou část funkce $f(x_1) = x^2$.*
- *V druhém kroku hledáme kombinaci hodnot x_1 a x_2, pro které bude hodnota $f(x_1, x_2)$ minimální, tedy* $\min f(x_1) + f(x_2)$ *pro $x_1 + x_2 = b$*

  Výpočty pro první tři hodnoty b – výsledek vypadá následovně: Tučně zvýrazněná hodnota znamená minimální (optimální) kombinaci. Takto dopočítáme pro všechna b – ostatní mezivýpočty v příloze č. **??**

x_1	x_2	F_{x1,x_2}	x_1	x_2	F_{x_1,x_2}	x_1	x_2	F_{x_1,x_2}
0	0,1	0,3162	0	0,2	0,4472	0	0,3	0,5477
0,1	0	**0,0100**	0,1	0,1	0,3262	0,1	0,2	0,4572
			0,2	0	**0,0400**	0,2	0,1	0,3562
						0,3	0	**0,0900**

- *V třetím kroku hledáme kombinaci vybraných hodnot kroku 2 s hodnotami pro x_3*

b	x_1	$F(x_1)$		x_2	$F(x_2)$	x_3	$F(x_3)$
0,0	0,0	0,0000		0,0	0,0000		
0,1	0,1	0,0100		0,0	0,0100		
0,2	0,2	0,0400		0,0	0,0400		
0,3	0,3	0,0900		0,0	0,0900		
0,4	0,4	0,1600		0,0	0,1600		
0,5	**0,5**	0,2500	←	**0,0**	0,2500		
0,6	0,6	0,3600		0,0	0,3600		
0,7	0,7	0,4900		0,0	0,4900		
0,8	0,8	0,6400		0,0	0,6400		
0,9	0,9	0,8100		0,0	0,8100		
1,0	1,0	1,0000		0,7	0,9267	**0,5**	0,4268

Tab. 7. Tabulka výpočtů – dynamické programování.

Stanovení výsledků – z taulky vyplynulo optimální $x_3 = 0,5$ a pro zbývající $b_k = 0,5$ je optimální kombinace $x_2 = 0$ a $x_1 = 0,5$

Určíme hodnotu $f_{min}(x_1, x_2, x_3)$*:*
$f(x_1, x_2, x_3) = x_1^2 + \sqrt{x_2} + x_3^2 sin\left(x_3\frac{\pi}{2}\right) = 0,5^2 + \sqrt{0} + 0,5^2 sin\left(0,5\frac{\pi}{2}\right) =$
$= 0,25 + 0,1768 = 0,4268$
Provedeme kontrolu limitů:
$x_1 + x_2 + x_3 = 0,5 + 0 + 0,5 = 1$
Limit byl tedy vyčerpán beze zbytku.

Mezivýsledky pro proměnnou x_3 nejsou součástí tabulky – pro pevně daný limit je nepotřebujeme. Ale v příloze č. (**??**) jsou uvedeny. A to z důvodu možnosti hlubší analýzy daného problému. Pokud snížíme limit, není problém dohledat potřebnou hodnotu a řešit příklad okamžitě.

Například pokud se sníží limit $x_1 + x_2 + x_3 = 1$ na $0,7$, najdeme v příloze č. (**??**) – tabulka $x_3 \in (0; 0,7)$ pro tuto hodnotu optimum $x_3 = 0,4$, $b_k - x_3 = 0,3$ a zpětným dohledáním dostaneme hodnoty $x_1 = 0,3$, $x_2 = 0$, $x_3 = 0,4$ s výslednou hodnotou $0,1840$.

Po navýšení limitu by se samozřejmě musely dopočítat hodnoty pro x_1 a x_2 a nový výpočet pro x_3.

II. PRAKTICKÁ ČÁST

V této části budou provedeny výpočty vzorových příkladů. Výpočty budou provedeny pomocí algoritmů zpracovaných v programu Mathematica, včetně komentáře.

2.4 Kvadratické programování

Byla zvolena úloha, na níž bude ukázáno zpracování pomocí obou metod kvadratického programování, uvedených v této práci.

Hledáme minimum funkce

$$f(\bar{x} = (x_1 - x_2 - 1)^2 + 3(x_2 - 4)^2 = x_1^2 - 2x_1 - 2x_1x_2 + 4x_2^2 - 22x_2 + 49$$
$$\min f(x_1, x_2) = \min(x_1^2 - 2x_1 - 2x_1x_2 + 4x_2^2 - 22x_2 + 49)$$

Při omezeních:

$x_1 + 2x_2 \leqq 6$

$4x_1 + x_2 \leqq 10$

a splněných podmínkách nezápornosti: $x_1, x_2 \geqq 0$

Příprava podkladů pro zpracování

Pro použití obou optimalizačních metod budeme potřebovat zápis v maticové formě.

Dosazením do zápisu funkce

$$f(\bar{x}) = a + \bar{c}^T\bar{x} + \frac{1}{2}\bar{x}^T\mathrm{Q}\bar{x} \tag{62}$$

Dostaneme:

$$f(x_1, x_2) = 49 + \begin{pmatrix} x_1 & x_2 \end{pmatrix}\begin{pmatrix} -2 \\ -22 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} x_1 & x_2 \end{pmatrix}\begin{pmatrix} 2 & -2 \\ -2 & 8 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \tag{63}$$

Podobně použijeme maticový zápis omezení:

$$\mathrm{A}\bar{x} = \begin{pmatrix} 1 & 2 \\ 4 & 1 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 6 \\ 10 \end{pmatrix} \tag{64}$$

Máme tedy:

$$\mathrm{A} = \begin{pmatrix} 1 & 2 \\ 4 & 1 \end{pmatrix} \qquad \mathrm{Q} = \begin{pmatrix} 2 & -2 \\ --2 & 8 \end{pmatrix} \tag{65}$$

$$\bar{b} = \begin{pmatrix} 6 \\ 10 \end{pmatrix} \qquad \bar{c} = \begin{pmatrix} -2 & -22 \end{pmatrix} \tag{66}$$

Pro potřeby naplnění simplexových tabulek dosadíme:

$$\mathrm{M} = \begin{pmatrix} \mathrm{Q} & \mathrm{A}^T \\ -\mathrm{A} & \bar{0} \end{pmatrix} = \begin{pmatrix} -2 & 2 & -1 & -4 \\ 2 & -8 & -2 & -1 \\ 1 & 2 & 0 & 0 \\ 4 & 1 & 0 & 0 \end{pmatrix} \tag{67}$$

Vektor $\bar{z}$

$$\bar{z} = \begin{pmatrix} \bar{c}^T \\ \bar{b} \end{pmatrix} = \begin{pmatrix} -2 \\ -22 \\ 6 \\ 10 \end{pmatrix} \tag{68}$$

2.4.1 Výpočet pomocí metody Shetty-Lemke

Program začíná definicemi pomocných matic:

```
pb = {{"w1"}, {"w2"}, {"w3"}, {"w4"}};(* vektor pb*)
m = {{"z1"}, {"z2"}, {"z3"}, {"z4"}};(* vektor z*)
pom = {{0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}};
pom1 = {{0}, {0}, {0}, {0}};
(* konec definice pomocných matic*)
```

Dalším krokem je naplnění matic

```
q = {{-2}, {-22}, {6}, {10}};(* vektor q*)
y = {{0}, {0}, {0}, {0}};(* vektor q*)
w = {{1, 0, 0, 0},
     {0, 1, 0, 0},
     {0, 0, 1, 0},
     {0, 0, 0, 1}};(* vektor w*)
z = {{-2, 2, -1, -4},
     {2, -8, -2, -1},
     {1, 2, 0, 0},
     {4, 1, 0, 0}};(* vektor z*)
```

Sestavení matice pro simplexovou tabulku a její kontrolní tisk:

```
s = Transpose[{pb[[All, 1]], q[[All, 1]], w[[All, 1]], w[[All, 2]],
    w[[All, 3]], w[[All, 4]], z[[All, 1]], z[[All, 2]], z[[All, 3]],
    z[[All, 4]], y[[All, 1]]}];
MatrixForm[s]
```

$$\begin{pmatrix} w1 & -2 & 1 & 0 & 0 & 0 & -2 & 2 & -1 & -4 & 0 \\ w2 & -22 & 0 & 1 & 0 & 0 & 2 & -8 & -2 & -1 & 0 \\ w3 & 6 & 0 & 0 & 1 & 0 & 1 & 2 & 0 & 0 & 0 \\ w4 & 10 & 0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Naplnění pomocné proměnné y, výběr vhodného kandidáta (řádek 2), jeho přechod mezi bázické a dopočítání ostatních řádků s $y = -1$ (řádek 1). Poté opět kontrolní tisk.

```
For[ypos = 1, ypos < 4, ypos++,
 If[s[[ypos, 2]] < 0, s[[ypos, 11]] = -1,]
 ]
MatrixForm[s](* matice s~- úprava y*)
minq = Position[q, Min[q]];
minqy = minq[[1, 1]];
selxpos = minqy + 6;
(*Zaměření pozice pro y*)
ss = s;
For[ypos = 1, ypos < 5, ypos++,
 If[ypos == minqy, ss[[ypos, All]] = ss[[ypos, All]]*(-1);
  ss[[ypos, 1]] = "y",]]
(*přechod y mezi bázické*)
For[ypos = 1, ypos < 5, ypos++,
 pb = ss[[ypos, 1]];
 If[ss[[ypos, 11]] == (-1),
  ss[[ypos, All]] = ss[[ypos, All]] + ss[[minqy, All]];
  ss[[ypos, 1]] = pb,]
 ]
s = ss;
MatrixForm[s]
```

$$\begin{pmatrix} w1 & 20 & 1 & -1 & 0 & 0 & -4 & 10 & 1 & -3 & 0 \\ y & 22 & 0 & -1 & 0 & 0 & -2 & 8 & 2 & 1 & 1 \\ w3 & 6 & 0 & 0 & 1 & 0 & 1 & 2 & 0 & 0 & 0 \\ w4 & 10 & 0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Nyní začíná smyčka s podmínkou pro ukončení, pokud y vypadne z bázických proměnných (testování podmínky uzavírá cyklus). Následují pomocné matice pro výběr sloupce s minimální proměnnou p (kandidát z_t na přechod mezi bázické. Vzhledem k tomu, že hledáme $\min p$, jsou nevyhovující řešení označena hodnotou ∞.

```
testy=1;
While[testy>1,
For[ypos = 1, ypos < 5, ypos++,
 For[xpos = 7, xpos < 11, xpos++,
  If[s[[ypos, xpos]] > 0 && s[[ypos, xpos - 4]] != 0 &&
    StringTake[s[[ypos, 1]], {1}] != "z",
   pom[[ypos, xpos - 6]] = s[[ypos, 2]]/s[[ypos, xpos]],
   pom[[ypos, xpos - 6]] = \[Infinity]]
  ]
 ]
MatrixForm[pom]
minpos = Position[pom, Min[pom]];
selxpos = minpos[[1, 2]] + 6;
```

$$\begin{pmatrix} \infty & 2 & \infty & \infty \\ \infty & \frac{11}{4} & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \end{pmatrix}$$

Zúžení výběru na konkrétní hodnotu – vybíráme sloupec kanidáta z_t z předchozí matice a hledáme proměnnou $\min p_t$) pro dosazení mezi bázické proměnné (řádek).

```
For[ypos = 1, ypos < 5, ypos++,
 If[s[[ypos, selxpos]] == 0 || StringTake[s[[ypos, 1]], {1}] == "z",
  pom1[[ypos, 1]] = \[Infinity],
  pom1[[ypos, 1]] = s[[ypos, 2]]/s[[ypos, selxpos]]] ]
MatrixForm[pom1]
minpos = Position[pom1, Min[pom1]];
minposy = minpos[[1, 1]];
(*vyhodnocení podmínky min q/m=>i číslo řádku*)
MatrixForm[pom1]
```

$$\begin{pmatrix} 2 \\ \frac{11}{4} \\ \infty \\ \infty \end{pmatrix}$$

Následují poslední kroky výpočtů – dosazení vybrané proměnné z_t mezi bázické, dopočítání vybraného řádku a přepočet ostatních řádků podle vybraného klíče (zde v prvním kroku $\frac{11}{4}$)

```
For[ypos = 1, ypos < 5, ypos++,
 If[ypos == minposy,
  ss[[ypos, All]] = ss[[ypos, All]]/ss[[minposy, selxpos]];
  ss[[ypos, 1]] = m[[selxpos - 6, 1]],] ]
(*přechod vybrané z~mezi bázické*)
For[ypos = 1, ypos < 5, ypos++,
 If[ypos != minposy,
  For[xpos = 2, xpos < 11, xpos++,
   ss[[ypos, xpos]] =
     s[[ypos, xpos]] - s[[ypos, selxpos]]*ss[[minposy, xpos]];
   ],] ]
(*dopočítání zbytku tabulky*)
s = ss;
testy=0;
For[ypos = 1,ypos < 5,ypos++,
 If[StringTake[s[[ypos, 1]], {1}] == "y",testy=testy+1,]
]
(*testovací smyčka na existenci "y" mezi bázickými*)
](*konec While*)
MatrixForm[s]
```

Výsledek po prvním kroku:

$$\begin{pmatrix} z2 & 2 & \frac{1}{10} & -\frac{1}{10} & 0 & 0 & -\frac{2}{5} & 1 & \frac{1}{10} & -\frac{3}{10} & 0 \\ y & 6 & -\frac{4}{5} & -\frac{1}{5} & 0 & 0 & \frac{6}{5} & 0 & \frac{6}{5} & \frac{17}{5} & 1 \\ w3 & 2 & -\frac{1}{5} & \frac{1}{5} & 1 & 0 & \frac{9}{5} & 0 & -\frac{1}{5} & \frac{3}{5} & 0 \\ w4 & 8 & -\frac{1}{10} & \frac{1}{10} & 0 & 1 & \frac{22}{5} & 0 & -\frac{1}{10} & \frac{3}{10} & 0 \end{pmatrix}$$

Z dalších kroků jen výsledky:

Krok 2

$$\begin{pmatrix} \infty & \infty & \infty & \infty \\ 5 & \infty & \infty & \infty \\ \frac{10}{9} & \infty & \infty & \infty \\ \frac{20}{11} & \infty & \infty & \frac{80}{3} \end{pmatrix} \quad \begin{pmatrix} \infty \\ 5 \\ \frac{10}{9} \\ \frac{20}{11} \end{pmatrix} \quad \begin{pmatrix} z2 & \frac{22}{9} & \frac{1}{18} & -\frac{1}{18} & \frac{2}{9} & 0 & 0 & 1 & \frac{1}{18} & -\frac{1}{6} & 0 \\ y & \frac{14}{3} & -\frac{3}{2} & -\frac{1}{3} & -\frac{2}{3} & 0 & 0 & 0 & \frac{4}{3} & 3 & 1 \\ z1 & \frac{10}{9} & -\frac{1}{9} & \frac{1}{9} & \frac{5}{9} & 0 & 1 & 0 & -\frac{1}{9} & \frac{1}{3} & 0 \\ w4 & \frac{28}{9} & -\frac{7}{18} & -\frac{7}{18} & -\frac{22}{9} & 1 & 0 & 0 & \frac{7}{18} & -\frac{7}{6} & 0 \end{pmatrix}$$

Krok 3

$$\begin{pmatrix} \infty & \infty & \infty & \infty \\ \infty & \infty & \frac{7}{2} & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & 8 & \infty \end{pmatrix} \quad \begin{pmatrix} \infty \\ \frac{7}{2} \\ \infty \\ 8 \end{pmatrix} \quad \begin{pmatrix} z2 & \frac{9}{4} & \frac{1}{12} & -\frac{1}{24} & \frac{1}{4} & 0 & 0 & 1 & 0 & -\frac{7}{24} & 0 \\ z3 & \frac{7}{2} & -\frac{1}{2} & -\frac{1}{4} & -\frac{1}{2} & 0 & 0 & 0 & 1 & \frac{9}{4} & \frac{3}{4} \\ z1 & \frac{3}{2} & -\frac{1}{6} & \frac{1}{12} & \frac{1}{2} & 0 & 1 & 0 & 0 & \frac{7}{12} & 0 \\ w4 & \frac{7}{4} & \frac{7}{12} & -\frac{7}{24} & -\frac{9}{4} & 1 & 0 & 0 & 0 & -\frac{49}{24} & 0 \end{pmatrix}$$

Z bázických vypadla proměnná y, takže smyčka While byla ukončena a z poslední dopočítané tabulky můžeme odečíst výsledky:

$x_1 = \frac{3}{2} \qquad x_2 = \frac{9}{4} \qquad \lambda_1 = \frac{7}{2} \qquad \lambda_2 = 0$

Hodnotu účelové funkce získáme dosazením: $f(x_1, x_2) = f\left(\frac{3}{2}, \frac{9}{4}\right) = \frac{49}{4}$

Ověření výsledků

Provedeme ověření platnosti omezení.

$x_1 + 2x_2 \leqq 6 \quad \rightarrow \frac{3}{2} + 2\frac{9}{4} \leqq 6 \quad \rightarrow \frac{6}{4} + \frac{18}{4} \leqq 6 \quad \rightarrow \frac{24}{4} \leqq 6 \quad \rightarrow 6 \leqq 6$

$4x_1 + x_2 \leqq 10 \quad \rightarrow 4\frac{3}{2} + \frac{9}{4} \leqq 10 \quad \rightarrow \frac{24}{4} + \frac{9}{4} \leqq 10 \quad \rightarrow \frac{35}{4} \leqq 10 \quad \rightarrow 8,75 \leqq 10$

$x_1, x_2 \geqq 0 \qquad x_1 = \frac{3}{2} \quad \rightarrow \frac{3}{2} \geqq 0\,; \quad x_2 = \frac{9}{4} \quad \rightarrow \frac{9}{4} \geqq 0$

Výsledek $x_1 = \frac{3}{2}$ a $x_2 = \frac{9}{4}$ tedy splňují všechny podmínky.

2.4.2 Výpočet pomocí metody Whinston, van de Panne

Vzhledem k podobným východiskům jsou u obou programů shodné počáteční definice matic:

Začínáme definicemi pomocných matic:

```
pb = {{"w1"}, {"w2"}, {"w3"}, {"w4"}};(* vektor pb*)
m = {{"z1"}, {"z2"}, {"z3"}, {"z4"}};(* vektor pb*)
pom = {{0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}};
pom1 = {{0}, {0}, {0}, {0}};
(* konec definice pomocných matic*)
```

Pokračujeme naplněním matic zadání.

```
q = {{-2}, {-22}, {6}, {10}};(* vektor q*)
w = {{1, 0, 0, 0},
     {0, 1, 0, 0},
     {0, 0, 1, 0},
     {0, 0, 0, 1}};(* vektor w*)
z = {{-2, 2, -1, -4},
     {2, -8, -2, -1},
     {1, 2, 0, 0},
     {4, 1, 0, 0}};(* vektor z*)
```

Sestavení simplexové tabulky a její kontrolní výstup:

```
s = Transpose[{pb[[All, 1]], q[[All, 1]], w[[All, 1]], w[[All, 2]],
    w[[All, 3]], w[[All, 4]], z[[All, 1]], z[[All, 2]], z[[All, 3]],
    z[[All, 4]]}];
MatrixForm[s](* matice s~- naplněná simplexová tabulka*)
```

$$\begin{pmatrix} w1 & -2 & 1 & 0 & 0 & 0 & -2 & 2 & -1 & -4 \\ w2 & -22 & 0 & 1 & 0 & 0 & 2 & -8 & -2 & -1 \\ w3 & 6 & 0 & 0 & 1 & 0 & 1 & 2 & 0 & 0 \\ w4 & 10 & 0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 \end{pmatrix}$$

Metoda Whinston-van de Panne vynechává prvotní výpočet proměnné y a prvním krokem je hned výběr vhodného kandidáta. Probíhá opět ve dvou krocích:

První určuje sloupec. Vybíráme minimální hodnotu (podmínka výběr záporné, která má nejvyšší absolutní hodnotu je nahrazena hledáním nejmenší mezi zápornými), takže nevhodní kandidáti jsou ohodnoceni ∞:

```
For[ypos = 1, ypos < 5,
 For[xpos = 7, xpos < 11,
  If[s[[ypos, xpos]] < 0 && s[[ypos, xpos - 4]] == 1 &&
    StringTake[s[[ypos, 1]], {1}] != "z",
   pom[[ypos, xpos - 6]] = s[[ypos, xpos]],
   pom[[ypos, xpos - 6]] = \[Infinity] ];
```

```
   xpos++];
 ypos++]
MatrixForm[pom]
minpos = Position[pom, Min[pom]];
selectposx1 = minpos[[1, 2]];
selectposx = minpos[[1, 2]] + 6;
(*výběr sloupce*)
```

$$\begin{pmatrix} -2 & \infty & \infty & \infty \\ \infty & -8 & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \end{pmatrix}$$

Z vybraného sloupce vybereme konkrétního kandidáta $\min p_j \left(\frac{\bar{q}_j}{\bar{m}_{js}} \right)$

```
For[ypos = 1, ypos < 5,
 If[s[[ypos, selectposx]] == 0 ||
    StringTake[s[[ypos, 1]], {1}] == "z" ||
    s[[ypos, 2]]/s[[ypos, selectposx]] < 0,
   pom1[[ypos, 1]] = \[Infinity],
   pom1[[ypos, 1]] = s[[ypos, 2]]/s[[ypos, selectposx]]]
  ypos++]
MatrixForm[pom1]
minpos = Position[pom1, Min[pom1]];
minposy = minpos[[1, 1]];
(*vyhodnocení podmínky min q/m => i číslo řádku*)
```

$$\begin{pmatrix} \infty \\ \frac{11}{4} \\ 3 \\ 10 \end{pmatrix}$$

Vybraného kandidáta převedeme mezi bázické a dopočítáme zbytek tabulky

```
ss = s;
For[ypos = 1, ypos < 5,
 If[ypos == minposy,
   ss[[ypos, All]] = ss[[ypos, All]]/ss[[minposy, selectposx]];
   ss[[ypos, 1]] = m[[selectposx1, 1]],] ypos++]
(*přechod vybrané z~mezi bázické*)
For[ypos = 1, ypos < 5,
 If[ypos != minposy,
    For[xpos = 2, xpos < 11,
     ss[[ypos, xpos]] =
      s[[ypos, xpos]] - s[[ypos, selectposx]]*ss[[minposy, xpos]];
     xpos++],]
   ypos++;]
(*dopočítání zbytku tabulky*)
s = ss;
```

Výsledek po prvním kroku:

$$\begin{pmatrix} w1 & -\frac{15}{2} & 1 & \frac{1}{4} & 0 & 0 & -\frac{3}{2} & 0 & -\frac{3}{2} & -\frac{17}{4} \\ z2 & \frac{11}{4} & 0 & -\frac{1}{8} & 0 & 0 & -\frac{1}{4} & 1 & \frac{1}{4} & \frac{1}{8} \\ w3 & \frac{1}{2} & 0 & \frac{1}{4} & 1 & 0 & \frac{3}{2} & 0 & -\frac{1}{2} & -\frac{1}{4} \\ w4 & \frac{29}{4} & 0 & \frac{1}{8} & 0 & 1 & \frac{17}{4} & 0 & -\frac{1}{4} & -\frac{1}{8} \end{pmatrix}$$

Následuje ověření podmínky pro ukončení smyčky a po posledním kroku následuje tisk výsledku.

Mezivýsledky po druhém kroku:

$$\begin{pmatrix} -\frac{3}{2} & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & -\frac{1}{2} & \infty \\ \infty & \infty & \infty & -\frac{1}{8} \end{pmatrix} \quad \begin{pmatrix} 5 \\ \infty \\ \frac{1}{3} \\ \frac{29}{17} \end{pmatrix} \quad \begin{pmatrix} w1 & -7 & 1 & \frac{1}{2} & 1 & 0 & 0 & 0 & -2 & -\frac{9}{2} \\ z2 & \frac{17}{6} & 0 & -\frac{1}{12} & \frac{1}{6} & 0 & 0 & 1 & \frac{1}{6} & \frac{1}{12} \\ z1 & \frac{1}{3} & 0 & \frac{1}{6} & \frac{2}{3} & 0 & 1 & 0 & -\frac{1}{3} & -\frac{1}{6} \\ w4 & \frac{35}{6} & 0 & -\frac{7}{12} & -\frac{17}{6} & 1 & 0 & 0 & \frac{7}{6} & \frac{7}{12} \end{pmatrix}$$

Mezivýsledky po třetím kroku:

$$\begin{pmatrix} \infty & \infty & -2 & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \end{pmatrix} \quad \begin{pmatrix} \frac{7}{2} \\ \infty \\ \infty \\ 5 \end{pmatrix} \quad \begin{pmatrix} z3 & \frac{7}{2} & -\frac{1}{2} & -\frac{1}{4} & -\frac{1}{2} & 0 & 0 & 0 & 1 & \frac{9}{4} \\ z2 & \frac{9}{4} & \frac{1}{12} & -\frac{1}{24} & \frac{1}{4} & 0 & 0 & 1 & 0 & -\frac{7}{24} \\ z1 & \frac{3}{2} & -\frac{1}{6} & \frac{1}{12} & \frac{1}{2} & 0 & 1 & 0 & 0 & \frac{7}{12} \\ w4 & \frac{7}{4} & \frac{7}{12} & -\frac{7}{24} & -\frac{9}{4} & 1 & 0 & 0 & 0 & -\frac{49}{24} \end{pmatrix}$$

Pro další krok neexistuje vhodný kandidát na přechod mezi bázické, takže program je ukončen.

Výsledné hodnoty jsou stejně jako u metody Shetty-Lemke:

$x_1 = \frac{3}{2} \qquad x_2 = \frac{9}{4} \qquad \lambda_1 = \frac{7}{2} \qquad \lambda_2 = 0$

Hodnotu účelové funkce získáme dosazením: $f(x_1, x_2) = f\left(\frac{3}{2}, \frac{9}{4}\right) = \frac{49}{4}$

Ověření výsledků

Je počítán stejný příklad jako u předchozí metody a bylo dosaženo stejných výsledků. Platí tedy i ověření výsledků z příkladu počítaného metodou Shetty-Lemke.

2.5 Dynamické programování

Zadání úlohy:

hledejte min $f(\bar{x}) = \sin(2x_1) + \tan(x_2) + \frac{4}{\pi}x_3$

Při omezeních:

$x_1 + x_2 + x_3 = \frac{\pi}{4}$

$x_1, x_2, x_3 \geqq 0$

A daném kroku $h = \frac{\pi}{40}$

Příprava podkladů

Zde je příprava podkladů o poznání jednodušší. Jedna ze základních vlastností u úloh dynamického programování je separovatelnost účelové funkce. $f(\bar{x})$ rozdělíme:

$f(x_1, x_2, x_3) = F(x_1, x_2) + F(x_3)$

odtud tedy:

$F(x_1, x_2) = \sin(2x_1) + \tan(x_2), \qquad F(x_3) = \frac{4}{\pi}x_3$

dále máme určený limit $\frac{\pi}{4}$ a krok $\frac{\pi}{40}$.

Tím jsme určili všechny proměnné, které budeme potřebovat pro zadání úlohy.

Program kopíruje výpočetní postup pro „ruční“ metodu popsanou v kapitole (2.3.4). Začíná definicemi proměnných a matic:

```
h = \[Pi]/40;
limit = \[Pi]/4;
pocetcyklu = Round[limit/h];
Fx1x2 = Array[0 &, {pocetcyklu + 1, 3}];
Fx3 = Array[0 &, {pocetcyklu + 1, 3}];
Fx1x2[[1, All]] = 0;
```

Kde h je délka kroku, limit je hodnota z pravé strany funkce vyjadřující omezení. Dále se počítá počet kroků, na základě kterého se definuje rozsah matic pro uložení výsledků.

Následuje dvojitá (vnořená) smyčka s předpočteným počtem kroků. Vnější postupuje podle počtu kroků h v rámci limitu, vnitřní vytváří matici s hodnotami pro jednotlivé kombinace x_1 a x_2 v rámci aktuální počítané hodnoty $(h \cdot cykl)$. Hodnoty ukládá do pomocné matice. Její rozměr je definovaný v začátku cyklu (po vyjádření počtu kroků vnitřního cyklu).

Po ukončení vnitřního cyklu je v matici vyhledána minimální hodnota a uložena do finální matice $Fx1x2$ spolu s hodnotami odpovídající kombinace x_1 a x_2. Po ukončení vnějšího cyklu je tato matice pro názornost zobrazena.

```
 For[cykl = 1, cykl < pocetcyklu + 1, cykl++,
 dims = {(cykl + 1), 3};
 pom = Array[0 &, dims];
 MatrixForm[pom];
 For[b = 1, b < cykl + 2, b++,
  x1 = (b - 1)*h;
  x2 = cykl*h - x1;
  pom[[b, 1]] = x1;
  pom[[b, 2]] = x2;
  pom[[b, 3]] = Sin[2* x1] + Tan [x2];
  ];
 MatrixForm[pom];
 (*Print[MatrixForm[pom]];*)
 Fx1x2[[cykl + 1, 3]] = Min[pom[[All, 3]]];
 Fx1x2[[cykl + 1, 1]] = x1;
 Fx1x2[[cykl + 1, 2]] = x2;
 ]
Print[ "Minima za jednotlivé mezivýpočty, 1, sloupec x1, druhý \
sloupec x2, třetí sloupec výsledná hodnota Fx1x2"];
MatrixForm[Fx1x2]
For[cykl = 1, cykl < pocetcyklu + 2, cykl++,
 x12 = (cykl - 1)*h;
 x3 = limit - x12;
 Fx3[[cykl, 1]] = x12;
 Fx3[[cykl, 2]] = x3;
 Fx3[[cykl, 3]] = Fx1x2[[cykl, 3]] + 4*\[Pi]*x3;
 ]
Print[ "Minima za jednotlivé mezivýpočty, 1, sloupec bk, druhý \
sloupec x3, třetí sloupec výsledná hodnota Fx1x2+Fx3"];
MatrixForm[Fx3]
```

$$\begin{pmatrix} 0 & 0 & 0 \\ \frac{\pi}{40} & 0 & \text{Tan}\left[\frac{\pi}{40}\right] \\ \frac{\pi}{20} & 0 & \text{Tan}\left[\frac{\pi}{20}\right] \\ \frac{3\pi}{40} & 0 & \text{Tan}\left[\frac{3\pi}{40}\right] \\ \frac{\pi}{10} & 0 & \sqrt{1-\frac{2}{\sqrt{5}}} \\ \frac{\pi}{8} & 0 & \text{Tan}\left[\frac{\pi}{8}\right] \\ \frac{3\pi}{20} & 0 & \text{Tan}\left[\frac{3\pi}{20}\right] \\ \frac{7\pi}{40} & 0 & \text{Tan}\left[\frac{7\pi}{40}\right] \\ \frac{\pi}{5} & 0 & \sqrt{5-2\sqrt{5}} \\ \frac{9\pi}{40} & 0 & \text{Tan}\left[\frac{9\pi}{40}\right] \\ \frac{\pi}{4} & 0 & 1 \end{pmatrix}$$

Následuje druhá smyčka, tentokrát jednoduchá, která postupně načítá kombinaci hodnoty $b_k = x12$ a jejího doplňku $x3$. Vypočítá výslednou hodnotu $Fx1x2 + Fx3$. Ukládá je do finální matice, ze které potom vyhodnotí minimum. V matici je rovnou uložena i informace o b_k. Výsledek je potom kombinací hodnot z matice $Fx3$ (hodnota $x3$) a pro příslušnou $b_k = x12$ z matice $Fx1x2$.

$$\begin{pmatrix}
0 & \frac{\pi}{4} & \pi^2 \\
\frac{\pi}{40} & \frac{9\pi}{40} & \frac{9\pi^2}{10} + \mathrm{Tan}\left[\frac{\pi}{40}\right] \\
\frac{\pi}{20} & \frac{\pi}{5} & \frac{4\pi^2}{5} + \mathrm{Tan}\left[\frac{\pi}{20}\right] \\
\frac{3\pi}{40} & \frac{7\pi}{40} & \frac{7\pi^2}{10} + \mathrm{Tan}\left[\frac{3\pi}{40}\right] \\
\frac{\pi}{10} & \frac{3\pi}{20} & \sqrt{1 - \frac{2}{\sqrt{5}}} + \frac{3\pi^2}{5} \\
\frac{\pi}{8} & \frac{\pi}{8} & \frac{\pi^2}{2} + \mathrm{Tan}\left[\frac{\pi}{8}\right] \\
\frac{3\pi}{20} & \frac{\pi}{10} & \frac{2\pi^2}{5} + \mathrm{Tan}\left[\frac{3\pi}{20}\right] \\
\frac{7\pi}{40} & \frac{3\pi}{40} & \frac{3\pi^2}{10} + \mathrm{Tan}\left[\frac{7\pi}{40}\right] \\
\frac{\pi}{5} & \frac{\pi}{20} & \sqrt{5 - 2\sqrt{5}} + \frac{\pi^2}{5} \\
\frac{9\pi}{40} & \frac{\pi}{40} & \frac{\pi^2}{10} + \mathrm{Tan}\left[\frac{9\pi}{40}\right] \\
\frac{\pi}{4} & 0 & 1
\end{pmatrix}$$

Nakonec jsou vypsány výsledné hodnoty:

```
"Výsledkem je kombinace:"
Print["x1=", x1 = Fx1x2[[posxfin, 1]]];
Print["x2=", x2 = Fx1x2[[posxfin, 2]]];
Print["x3=", x3 = Fx3[[posxfin, 2]]];
"S výslednou hodnotou účelové funkce:"
 Print["f(x1,x2,x3)=", Fx3[[posxfin, 3]]];
Print["Dosazení do limitu: x1+x2+x3=", x1, "+", x2, "+", x3, "=",
 x1 + x2 + x3]
```

tedy:

$x_1 = x1 = \dfrac{\pi}{10}$

$x_2 = x2 = 0$

$x_3 = x3 = \dfrac{3\pi}{20}$

Výsledná hodnota účelové funkce $= \sqrt{1 - \dfrac{2}{\sqrt{5}}} + \dfrac{3\pi^2}{5} = 6,2467$

Ověření výsledků

Omezení $x_1 + x_2 + x_3 = \dfrac{\pi}{10} + 0 + \dfrac{3\pi}{20} = \dfrac{\pi}{4}$ byla dodržena. Limit byl zcela vyčerpán.

ZÁVĚR

Cílem této práce bylo vypracovat přehled optimalizačních metod, včetně jejich popisu a rozboru, používaných k řešení úloh neklasického vázaného extrému. Zejména pak ve smyslu konvexního, kvadratického a dynamického programování.

V úvodu teoretcké části byly zopakovány základní pojmy z oblasti statické optimalizace.

Teoretická část dále obsahuje rozbor úloh matematického programování. Začíná popisem úloh volného extrému, reprezentovaných zde metodou přípustných směrů. Pokračuje popisem klasického vázaného extrémemu a navazuje neklasickým vázaným extrémem s jednoduchou ukázkou úlohy lineárního programování, na které jsou vysvětleny pojmy „duální úloha“ a „citlivostní analýza“.

Následuje rozbor úloh nelineárního programování. Jsou klasifikovány podle matematických prinicipů, které využívají. Pokračuje rozbor vybraných metod na jednoduchých příkladech.

Konvexní programování je zde zastoupeno analytickou metodou postavenou na šesti Kuhn-Tuckerových podmínkách. Tato metoda se jeví jako poměrně složitá a špatně algoritmizovatelná. Proto se využívá dalších metod, jak využít vlastnosti konvexní množiny, kterou tvoří omezení.

Jednou z velmi používaných způsobů vycházející z konvexního programováníje kvadratické programování. Tyto úlohy mají využití nejen samy o sobě, ale mohou být využity i pro některé další úlohy nelineárního programování. Dosahuje se toho převedením obecného problému na posloupnost problémů kvadratických. Tato práce popisuje principy metod „Shetty-Lemke“ a „Whinston-van de Panne“. U obou je zpracována ukázková úloha.

Další popisovanou úlohou nelineárního programování je dynamické programování. Využívá separovatelnost účelové funkce a opírá se o Bellmanův princip optimality. Je popsaná metoda nalezení optima v orientovaném síťovém grafu a tabulkovou metodou.

Praktická část se vrací k popisu metod kvadratického a dynamického programování.

V progamu Mathematica jsou zde zalgoritmizovány úlohy kvadratického programování, s výpočtem pomocí obou popisovaných metod. Kód je rozdělen na jednotlivé výpočetní a rozhodovací bloky, které jsou samostatně popsány a doplněny o mezivýsledky. Vzhledem k tomu, že je různými metodami zpracovávaná stejná úloha, je dosaženo stejných výsledků. Metoda Whinston-van de Panne se zde ukazuje jako jednodušší pro zpracování.

Úloha dynamického programování je analyzována stejným způsobem. Kód pro Mathematicu je rozčleněna popsán. Vzhledem k velkému množství výpočtů (viz přílohu pro úlohu zpracovanou v teoretické části) jsou uvedeny jen výsledné matice s výběrem optimálních hodnot.

Závěrem lze konstatovat, že popsané úlohy jsou dobře algoritmizovatelné, jen u dynamického programování narážíme na vyšší paměťovou náročnost.

CONCLUSION

The aim of the thesis was to draw up an overview of the optimization methods that are used for problem processing of the nonclassical constrained extremes, including description and analysis of the methods, particularly in terms of convex, quadratic and dynamic programming.

In the preamble of the theoretical section the basic terms from the field of static optimization are recapitulated.

The theoretical section also includes the analysis of the problems of mathematical programming. It begins with the account of nonconstrained extreme problems represented by the method of feasible directions. Then it proceeds to the classic constrained extreme, including a simple example of the linear programming problem in which the terms of dual problem and sensitivity analysis are explained.

The analysis of nonlinear programming problems follows. The problems are classified according to the mathematical principles they employ. The analysis of the selected methods follows. The convex programming is here represented by the analytical method based on the analysis of six Kuhn-Tucker conditions. As this method seems to be rather complicated and difficult to be converted into algorithm, another methods are employed to make the account of the characteristics of convex set that is restricted.

One of the methods frequently used resulting from the convex programming is quadratic programming. These problems are utilized not only within their field but they can be also used for some other nonlinear programming problems. That is achieved by transferring general problem to the sequence of quadratic problems. The thesis describes the principles of Shetty-Lemke and Whinston-van de Penne methods and includes exemplar problems.

Another example of nonlinear programming that the thesis deals with is dynamic programming. It makes use of a target function separability and is based on Bellman's principle of optimization. It is a method of optimum location in directed reticular graph and a table method.

The practical section goes back to the description of quadratic and dynamic programming methods.

In Mathematica programme the problems of quadratic programming are converted into algorithm with the calculation mode of the two methods described. The code is divided into the particular computational and alternative boxes which are described and supplied with intermediate data. Inasmuch as the same problem is processed by various methods, the equivalent results are reached. Whinston-van de Penne's method proves to be easier for processing.

The dynamic programming problem is analysed the same way. The code for Mathematica programme is segmented and described. With regard to a large number of results (see the supplement relating to the problem processed in the theoretical section) only the resultant matrices containing the optimum values are presented.

At the conclusion it can be stated that the described problems are easily converted into algorithms, taken into consideration that dynamic programming is memory intensive.

SEZNAM POUŽITÉ LITERATURY

[1] Maňas, Miroslav: *Optimalizační metody* 3. přepracované vydání, Praha: SNTL – nakladatelství technické literatury, 1979,

[2] Pelikán, Jan: *Diskrétní modely v operačním výzkumu* 1. vydání, Praha: Kamil Minařík – PROFESSIONAL PUBLISHING, 2001, ISBN: 80-86419-17-7

[3] Lachout, Petr: *Matematické programování – pracovní text k přednášce Optimalizace I*, [online] 26.9.2008 [cit. 3.4.2009] dostupné z: `http://www.karlin.mff.cuni.cz/~lachout/Vyuka/Optima1/081026-Opt-text.ps`

[4] Dupačová, Jitka, Lachout, Petr: *Matematické programování – pracovní text k přednášce Optimalizace I*, [online] 7.1.2008 [cit. 20.3.2009] dostupné z: `http://www.karlin.mff.cuni.cz/~lachout/Vyuka/U-Optima/U-opt-text-080107.ps`

[5] Štecha, Jan: *Optimální rozhodování a řízení*, [online] 1999 [cit. 12.4.2009] dostupné z: `http://dce.felk.cvut.cz/orr/orr.pdf`

[6] Lukšan, Ladislav: *Numerické optimalizační metody (Technical report No. 930)*, [online] prosinec 2005 [cit. 25.3.2009] dostupné z: `http://www.cs.cas.cz/~luksan/lekce4.pdf`

[7] Lukšan, Ladislav: *Matematické programování (Technical report No. 1043)*, [online] prosinec 2008 [cit. 25.3.2009] dostupné z: `http://www.cs.cas.cz/~luksan/lekce6.pdf`

[8] Boyd, Stephen, Vandenberghe, Lieven: *Convex Optimization*, [online] Sixth printing with corrections 2008, Cambridge, Cambridge University Press, 2004, ISBN 978-0-521-83378-3, dostupné z: `http://www.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf`

[9] Přispěvatelé Wikipedie: *Konvexní programování*, [online] Wikipedie, Otevřená encyklopedie, c2009, 12. 9. 2005, Datum poslední revize 5. 01. 2009, 09:45 UTC, [citováno 4. 4. 2009] dostupné z: `http://cs.wikipedia.org/wiki/Konvexni_programovani`

[10] Přispěvatelé Wikipedie:, *Nelineární programování*, [online] Wikipedie: Otevřená encyklopedie, c2009, Datum poslední revize 5. 01. 2009, 09:57 UTC, [citováno 10. 04. 2009] dostupné z: `http://cs.wikipedia.org/wiki/Nelinearni_programovani`

[11] Přispěvatelé Wikipedie:, *Vázaný extrém*, [online] Wikipedie: Otevřená encyklopedie, c2009, Datum poslední revize 19. 11. 2008, 11:31 UTC, [citováno 13. 04. 2009] dostupné z: `http://cs.wikipedia.org/wiki/Vazany_extrem`

[12] Přispěvatelé Wikipedie:, *Konvexnost a konkávnost funkce*, [online] Wikipedie: Otevřená encyklopedie, c2009, Datum poslední revize 14. 3. 2009, 19:52 UTC, [citováno 13. 04. 2009] dostupné z: `http://cs.wikipedia.org/wiki/Konvexni_funkce`

[13] Kopačka, Juraj:, *Metódy sekvenčného kvadratického programovania*, [Diplomová práca – online] Univerzita Komenského, Fakulta matematiky, fyziky a informatiky, Katedra aplikovanej matematiky a štatistiky. Školiteľ: doc. RNDr. Milan Hamala, CSc. Rok obhajoby: 2004, [citováno 27. 04. 2009] dostupné z: `http://pc2.iam.fmph.uniba.sk/studium/efm/diplomovky/2004/kopacka/index.html`

[14] Černický, Petr:, *Metódy rozšírených Lagrangeových funkcií v konvexnom programovaní*, [Diplomová práca – online] Univerzita Komenského, Fakulta matematiky, fyziky a informatiky, Katedra aplikovanej matematiky a štatistiky. Školiteľ: doc. RNDr. Milan Hamala, CSc. Rok obhajoby: 2004, [citováno 27. 04. 2009] dostupné z: `http://pc2.iam.fmph.uniba.sk/studium/efm/diplomovky/2004/cernicky/index.html`

[15] Gancárová, Martina:, *Konvexná analýza a mikroekonomická teória firmy*, [Diplomová práca – online] Univerzita Komenského, Fakulta matematiky, fyziky a informatiky, Katedra aplikovanej matematiky a štatistiky. Prof. RNDr. Pavol Brunovský, DrSc.. Rok obhajoby: 2003, [citováno 27. 04. 2009] dostupné z: `http://pc2.iam.fmph.uniba.sk/studium/efm/diplomovky/2003/gancarova/index.html`

[16] Hudzovič, Peter: *Optimalizácia*, Bratislava, Vydavateľstvo STU Bratislava, 2004, ISBN: 80-227-2072-0

[17] Tomšová, Marie:, *Nelineární programování*, [online] Ústav matematiky, Fakulta elektrotechniky a komunikačních technologií, Vysoké učení technické v Brně. 2005, [citováno 2. 05. 2009] dostupné z: `http://math.fce.vutbr.cz/~pribyl/workshop_2007/prispevky/Tomsova.rtf`

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

*	extrém funkce (index hvězdička, např. $\bar{x}^*$)
^{T}	označení transfomace (index T, např. matice transformovaná matice $\mathrm{Q} \rightarrow \mathrm{Q}^T$)
$^-$	vektor (např. $\bar{x}$ je vektor proměnných x, tedy $x_1, x_2, \ldots, x_n$)
$^\rightarrow$	směrový vektor (např. $\vec{s}$ – vektor gradientu funkce)
λ	označení neurčitého koeficientu (Lagrangeova multiplikátoru) [lambda]

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

x_1	x_2	F_{x1,x_2}
0	0,1	0,3162
0,1	0	**0,0100**

x_1	x_2	F_{x_1,x_2}
0	0,2	0,4472
0,1	0,1	0,3262
0,2	0	**0,0400**

x_1	x_2	F_{x_1,x_2}
0	0,3	0,5477
0,1	0,2	0,4572
0,2	0,1	0,3562
0,3	0	**0,0900**

x_1	x_2	F_{x_1,x_2}
0	0,4	0,6325
0,1	0,3	0,5577
0,2	0,2	0,4872
0,3	0,1	0,4062
0,4	0	**0,1600**

x_1	x_2	F_{x_1,x_2}
0	0,5	0,7071
0,1	0,4	0,6425
0,2	0,3	0,5877
0,3	0,2	0,5372
0,4	0,1	0,4762
0,5	0	**0,2500**

x_1	x_2	F_{x_1,x_2}
0	0,6	0,7746
0,1	0,5	0,7171
0,2	0,4	0,6725
0,3	0,3	0,6377
0,4	0,2	0,6072
0,4	0,1	0,5662
0,5	0	**0,3600**

x_1	x_2	F_{x_1,x_2}
0	0,7	0,8367
0,1	0,6	0,7846
0,2	0,5	0,7471
0,3	0,4	0,7225
0,4	0,3	0,7077
0,5	0,2	0,6972
0,6	0,1	0,6762
0,7	0	**0,4900**

x_1	x_2	F_{x_1,x_2}
0	0,8	0,8944
0,1	0,7	0,8467
0,2	0,6	0,8146
0,3	0,5	0,7971
0,4	0,4	0,7925
0,5	0,3	0,7977
0,6	0,2	0,8072
0,7	0,1	0,8062
0,8	0	**0,6400**

x_1	x_2	F_{x_1,x_2}
0	0,9	0,9487
0,1	0,8	0,9044
0,2	0,7	0,8767
0,3	0,6	0,8646
0,4	0,5	0,8671
0,5	0,4	0,8825
0,6	0,3	0,9077
0,7	0,2	0,9362
0,8	0,1	0,9562
0,9	0	**0,8100**

x_1	x_2	F_{x_1,x_2}
0	1,0	1,0000
0,1	0,9	0,9587
0,2	0,8	0,9344
0,3	0,7	**0,9267**
0,4	0,6	0,9346
0,5	0,5	0,9571
0,6	0,4	0,9925
0,7	0,3	1,0377
0,8	0,2	1,0872
0,9	0,1	1,1262
1,0	0	1,0000

$b_k - x_3$	x_3	$F_{b_k-x_3,x_3}$
0	0,1	**0,0016**
0,1	0	0,0100

x_1	x_2	F_{x_1,x_2}
0	0,2	0,0124
0,1	0,1	**0,0116**
0,2	0	0,0400

$b_k - x_3$	x_3	$F_{b_k-x_3,x_3}$
0	0,3	0,0409
0,1	0,2	**0,0224**
0,2	0,1	0,0416
0,3	0	0,0900

$b_k - x_3$	x_3	$F_{b_k-x_3,x_3}$
0	0,4	0,0940
0,1	0,3	**0,0509**
0,2	0,2	0,0524
0,3	0,1	0,0916
0,4	0	0,1600

$b_k - x_3$	x_3	$F_{b_k-x_3,x_3}$
0	0,5	0,1768
0,1	0,4	0,1040
0,2	0,3	**0,0809**
0,3	0,2	0,1024
0,4	0,1	0,1616
0,5	0	0,2500

$b_k - x_3$	x_3	$F_{b_k-x_3,x_3}$
0	0,6	0,2912
0,1	0,5	0,1868
0,2	0,4	0,1340
0,3	0,3	**0,1309**
0,4	0,2	0,1724
0,5	0,1	0,2516
0,6	0	0,3600

$b_k - x_3$	x_3	$F_{b_k-x_3,x_3}$
0	0,7	0,4366
0,1	0,6	0,3012
0,2	0,5	0,2168
0,3	0,4	**0,1840**
0,4	0,3	0,2009
0,5	0,2	0,2624
0,6	0,1	0,3614
0,7	0	0,4900

$b_k - x_3$	x_3	$F_{b_k-x_3,x_3}$
0	0,8	0,6087
0,1	0,7	0,4466
0,2	0,6	0,3312
0,3	0,5	0,2668
0,4	0,4	**0,2540**
0,5	0,3	0,2909
0,6	0,2	0,3724
0,7	0,1	0,4916
0,8	0	0,6400

$b_k - x_3$	x_3	$F_{b_k-x_3,x_3}$
0	0,9	0,8000
0,1	0,8	0,6187
0,2	0,7	0,4766
0,3	0,6	0,3812
0,4	0,5	**0,3368**
0,5	0,4	0,3440
0,6	0,3	0,4009
0,7	0,2	0,5024
0,8	0,1	0,6416
0,9	0	0,8100

$b_k - x_3$	x_3	$F_{b_k-x_3,x_3}$
0	1,0	1,0000
0,1	0,9	0,8100
0,2	0,8	0,6487
0,3	0,7	0,5266
0,4	0,6	0,4512
0,5	0,5	**0,4268**
0,6	0,4	0,4540
0,7	0,3	1,5309
0,8	0,2	1,6524
0,9	0,1	1,8116
1,0	0	1,9267

```
pb = {{"w1"}, {"w2"}, {"w3"}, {"w4"}};(* vektor pb*)
m = {{"z1"}, {"z2"}, {"z3"}, {"z4"}};(* vektor z*)
pom = {{0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}};
pom1 = {{0}, {0}, {0}, {0}};
(* konec definice pomocných matic*)
q = {{-2}, {-22}, {6}, {10}};(* vektor q*)
y = {{0}, {0}, {0}, {0}};(* vektor q*)
w = {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0,
    1}};(* vektor w*)
z = {{-2, 2, -1, -4}, {2, -8, -2, -1}, {1, 2, 0, 0}, {4, 1, 0,
    0}};(* vektor z*)
s = Transpose[{pb[[All, 1]], q[[All, 1]], w[[All, 1]], w[[All, 2]],
    w[[All, 3]], w[[All, 4]], z[[All, 1]], z[[All, 2]], z[[All, 3]],
    z[[All, 4]], y[[All, 1]]}];
MatrixForm[s](* matice s - naplněná simplexová tabulka*)
For[ypos = 1, ypos < 4, ypos++,
 If[s[[ypos, 2]] < 0, s[[ypos, 11]] = -1,]
 ]
MatrixForm[s](* matice s - úprava y*)
minq = Position[q, Min[q]];
minqy = minq[[1, 1]];
selxpos = minqy + 6;
(*Zaměření pozice pro y*)
ss = s;
For[ypos = 1, ypos < 5, ypos++,
 If[ypos == minqy, ss[[ypos, All]] = ss[[ypos, All]]*(-1);
  ss[[ypos, 1]] = "y",]]
(*přechod y mezi bázické*)
For[ypos = 1, ypos < 5, ypos++,
 pb = ss[[ypos, 1]];
 If[ss[[ypos, 11]] == (-1),
  ss[[ypos, All]] = ss[[ypos, All]] + ss[[minqy, All]];
  ss[[ypos, 1]] = pb,]
 ]
s = ss;
MatrixForm[s]
(*Smyčka*)
smycka = 1;
While[smycka == 1,
 For[ypos = 1, ypos < 5, ypos++,
   For[xpos = 7, xpos < 11, xpos++,
    If[s[[ypos, xpos]] > 0 && s[[ypos, xpos - 4]] != 0 &&
      StringTake[s[[ypos, 1]], {1}] != "z",
     pom[[ypos, xpos - 6]] = s[[ypos, 2]]/s[[ypos, xpos]],
     pom[[ypos, xpos - 6]] = \[Infinity]]
    ]
   ]
  MatrixForm[pom]
  minpos = Position[pom, Min[pom]];
 selxpos = minpos[[1, 2]] + 6;
 For[ypos = 1, ypos < 5, ypos++,
   If[s[[ypos, selxpos]] == 0 ||
     StringTake[s[[ypos, 1]], {1}] == "z",
    pom1[[ypos, 1]] = \[Infinity],
    pom1[[ypos, 1]] = s[[ypos, 2]]/s[[ypos, selxpos]]] ]
  MatrixForm[pom1]
  minpos = Position[pom1, Min[pom1]];
 minposy = minpos[[1, 1]];
 (*vyhodnocení podmínky min q/m=>i číslo řádku*)
 For[ypos = 1, ypos < 5, ypos++,
```

```
   If[ypos == minposy,
    ss[[ypos, All]] = ss[[ypos, All]]/ss[[minposy, selxpos]];
    ss[[ypos, 1]] = m[[selxpos - 6, 1]],] ]
  (*přechod vybrané z mezi bázické*)
  For[ypos = 1, ypos < 5, ypos++,
   If[ypos != minposy,
    For[xpos = 2, xpos < 11, xpos++,
     ss[[ypos, xpos]] =
       s[[ypos, xpos]] - s[[ypos, selxpos]]*ss[[minposy, xpos]];
    ],] ];
 (*dopočítání zbytku tabulky*)
 s = ss;
 smycka = 0;
 For[ypos = 1, ypos < 5, ypos++,
   If[StringTake[s[[ypos, 1]], {1}] == "y", smycka++,
    ]
   ]
  If[smycka >= 1, smycka = 1,
   ]
 ]
MatrixForm[s]
```

```
pb = {{"w1"}, {"w2"}, {"w3"}, {"w4"}};(* vektor pb*)
m = {{"z1"}, {"z2"}, {"z3"}, {"z4"}};(* vektor pb*)
pom = {{0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}};
pom1 = {{0}, {0}, {0}, {0}};
(* konec definice pomocných matic*)
q = {{-2}, {-22}, {6}, {10}};(* vektor q*)
w = {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0,
    1}};(* vektor w*)
z = {{-2, 2, -1, -4}, {2, -8, -2, -1}, {1, 2, 0, 0}, {4, 1, 0,
    0}};(* vektor z*)
s = Transpose[{pb[[All, 1]], q[[All, 1]], w[[All, 1]], w[[All, 2]],
    w[[All, 3]], w[[All, 4]], z[[All, 1]], z[[All, 2]], z[[All, 3]],
    z[[All, 4]]}];
MatrixForm[s](* matice s - naplněná simplexová tabulka*)
For[ypos = 1, ypos < 5,
 For[xpos = 7, xpos < 11,
  If[s[[ypos, xpos]] < 0 && s[[ypos, xpos - 4]] == 1 &&
    StringTake[s[[ypos, 1]], {1}] != "z",
   pom[[ypos, xpos - 6]] = s[[ypos, xpos]],
   pom[[ypos, xpos - 6]] = \[Infinity] ];
   xpos++];
 ypos++]
MatrixForm[pom];
minpos = Position[pom, Min[pom]];
selectposx1 = minpos[[1, 2]];
selectposx = minpos[[1, 2]] + 6;
(*výběr sloupce*)
For[ypos = 1, ypos < 5,
 If[s[[ypos, selectposx]] == 0 ||
    StringTake[s[[ypos, 1]], {1}] == "z" ||
    s[[ypos, 2]]/s[[ypos, selectposx]] < 0,
   pom1[[ypos, 1]] = \[Infinity],
   pom1[[ypos, 1]] = s[[ypos, 2]]/s[[ypos, selectposx]]]
  ypos++]
MatrixForm[pom1];
minpos = Position[pom1, Min[pom1]];
minposy = minpos[[1, 1]];
(*vyhodnocení podmínky min q/m => i číslo řádku*)
smycka = 1;
While[smycka == 1,
 ss = s;
 For[ypos = 1, ypos < 5,
   If[ypos == minposy,
     ss[[ypos, All]] = ss[[ypos, All]]/ss[[minposy, selectposx]];
     ss[[ypos, 1]] = m[[selectposx1, 1]],] ypos++]
  (*přechod vybrané z mezi bázické*)
  For[ypos = 1, ypos < 5,
   If[ypos != minposy,
      For[xpos = 2, xpos < 11,
       ss[[ypos, xpos]] =
        s[[ypos, xpos]] - s[[ypos, selectposx]]*ss[[minposy, xpos]];
       xpos++],]
     ypos++;];
 (*dopočítání zbytku tabulky*)
 s = ss;
 For[ypos = 1, ypos < 5,
   For[xpos = 7, xpos < 11,
    If[s[[ypos, xpos]] < 0 && s[[ypos, xpos - 4]] == 1 &&
      StringTake[s[[ypos, 1]], {1}] != "z",
```

```
     pom[[ypos, xpos - 6]] = s[[ypos, xpos]],
     pom[[ypos, xpos - 6]] = \[Infinity] ];
     xpos++];
   ypos++]
  MatrixForm[pom];
 minpos = Position[pom, Min[pom]];
 selectposx1 = minpos[[1, 2]];
 selectposx = minpos[[1, 2]] + 6;
 (*výběr sloupce*)
 For[ypos = 1, ypos < 5,
   If[s[[ypos, selectposx]] == 0 ||
      StringTake[s[[ypos, 1]], {1}] == "z" ||
      s[[ypos, 2]]/s[[ypos, selectposx]] < 0,
     pom1[[ypos, 1]] = \[Infinity],
     pom1[[ypos, 1]] = s[[ypos, 2]]/s[[ypos, selectposx]]]
    ypos++]
  MatrixForm[pom1];
 minpos = Position[pom1, Min[pom1]];
 minposy = minpos[[1, 1]];
 (*vyhodnocení podmínky min q/m => i číslo řádku*)
 If[pom1[[minposy, 1]] == \[Infinity], smycka = 2, ]
 (*kontrola smyčky*)
 ]
MatrixForm[s]
```

```
h = \[Pi]/40;
limit = \[Pi]/4;
pocetcyklu = Round[limit/h];
Fx1x2 = Array[0 &, {pocetcyklu + 1, 3}];
Fx3 = Array[0 &, {pocetcyklu + 1, 3}];
Fx1x2[[1, All]] = 0;
For[cykl = 1, cykl < pocetcyklu + 1, cykl++,
 dims = {(cykl + 1), 3};
 pom = Array[0 &, dims];
 MatrixForm[pom];
 For[b = 1, b < cykl + 2, b++,
  x1 = (b - 1)*h;
  x2 = cykl*h - x1;
  pom[[b, 1]] = x1;
  pom[[b, 2]] = x2;
  pom[[b, 3]] = Sin[2* x1] + Tan [x2];
  ];
 MatrixForm[pom];
 (*Print[MatrixForm[pom]];*)
 Fx1x2[[cykl + 1, 3]] = Min[pom[[All, 3]]];
 Fx1x2[[cykl + 1, 1]] = x1;
 Fx1x2[[cykl + 1, 2]] = x2;
 ]
Print[ "Minima za jednotlivé mezivýpočty, 1, sloupec x1, druhý \
sloupec x2, třetí sloupec výsledná hodnota Fx1x2"];
MatrixForm[Fx1x2]
For[cykl = 1, cykl < pocetcyklu + 2, cykl++,
 x12 = (cykl - 1)*h;
 x3 = limit - x12;
 Fx3[[cykl, 1]] = x12;
 Fx3[[cykl, 2]] = x3;
 Fx3[[cykl, 3]] = Fx1x2[[cykl, 3]] + 4*\[Pi]*x3;
 ]
Print[ "Minima za jednotlivé mezivýpočty, 1, sloupec bk, druhý \
sloupec x3, třetí sloupec výsledná hodnota Fx1x2+Fx3"];
MatrixForm[Fx3]
xfin = Position[Fx3, Min[Fx3[[All, 3]]]];
posxfin = xfin[[1, 1]];
"Výsledkem je kombinace:"
Print["x1=", x1 = Fx1x2[[posxfin, 1]]];
Print["x2=", x2 = Fx1x2[[posxfin, 2]]];
Print["x3=", x3 = Fx3[[posxfin, 2]]];
"S výslednou hodnotou účelové funkce:"
 Print["f(x1,x2,x3)=", Sin[2 x1] + Tan[x2] + 4/\[Pi]*x3];
Print["Dosazení do limitu: x1+x2+x3=", x1, "+", x2, "+", x3, "=",
 x1 + x2 + x3]
```

//
/Diplomová práce.pdf
/Podpora výuky.ppt
/Mathematica/ (soubory pro program Mathematica)
/dynamicke_programovani3.nb
/Shetty-Lemke-smycka.nb
/Winnston-van_de_Panne-smycka.nb
/Pracovní/
/_actual/ (Zdrojové soubory pro CSLaTeX, pracovní soubory)
/podklady/ (Elektronická podoba citovaných materiálů...)
/obal/ (podklady pro tisk obalu Diplomové práce)
/_prezentace (pracovní materiály pro prezentaci Podpora výuky.ppt)
/Přílohy/ (tištěné přílohy diplomové práce)
/CD.pdf (obsah CD – tento dokument)
/dnp.pdf (mezivýpočty k příkladu dynamického programování – teoretická část)
/kod - dynamicke programovani.pdf (kód úlohy dynamického programování)
/kod - Shetty-Lemke.pdf (kód úlohy kvadratického programování – Shetty-Lemke)
/kod - Whinston-van de Panne.pdf (kód úlohy kvadratického programování – Whinston-van de Panne)