Celočíselné lineární programování

Přemysl Šůcha

11. března 2004

1 Úvod

Celá řada praktických problémů týkajících se optimalizace může být modelována a řešena pomocí integer (celočíselného[1] nebo také diskrétního) lineárního programování ILP (Integer Linear Programming). Tato úloha se od úlohy běžného lineárního programování LP liší vtom, že proměnné jsou omezeny na celá čísla. Pokud některé proměnné mohou nabývat i neceločíselných hodnot, potom se úloha nazývá smíšení celočíselné programování MIP (Mixed Integer Programming) [1]. Pro další použití označme množinu proměnných, které mohou nabývat pouze celočíselných hodnot X_Z.

Pokud bychom takovou úlohu řešili pomocí lineárního programování, s tím že bychom výsledek zaokrouhlili, nejenom že bychom neměli zaručeno že výsledné řešení bude optimální ale ani to, zda bude přípustné. Hlavní nevýhodou ILP je časová složitost algoritmu řešícího tuto úlohu. Zatímco úloha LP je řešitelná v polynomiálním čase, úloha ILP je tzv. NP-těžká (NP-hard) tzn. není znám polynomiální algoritmus. Mezi nejznámější metody řešení obecné úlohy ILP patří:

- Výčtové metody (Enumerative Methods)
- Metoda větví a mezí (Branch and Bound)
- Metody sečných nadrovin (Cutting Planes Methods)

Některé speciální případy jsou řešitelné pomocí polynomiálních algoritmů. Ty jsou založeny například na "hladovém" přístupu. To znamená že algoritmus se v každé iteraci rozhoduje tak aby co nejvíce maximalizoval cílovou funkci. Takové úlohy jsou pak obvykle řešeny převodem na odpovídající grafový algoritmus apod. [1, 2]). Jeden z nejznámějších problémů, kde lze použít hladový přístup je úloha minimální kostry [3, 1]. Bohužel podobné úlohy, pro které hladový postup dává zaručeně optimální výsledky, jsou dosti vzácné.

V celém textu předpokládejme problém maximalizace hodnoty cílové funkce. Problém minimalizace je možné na maximalizaci převést například obrácením znaménka u cílové funkce.

2 Výčtové metody

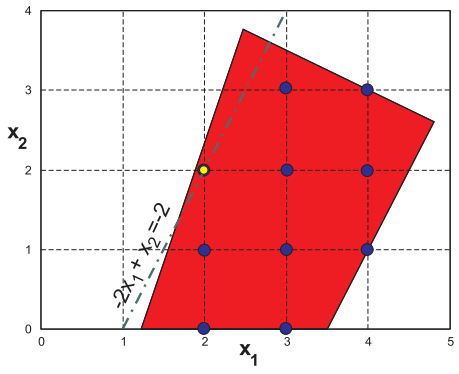
Výpočet je založen na prohledávání oblasti zahrnující všechna přípustná řešení [1, 2]. Vzhledem k celočíselnému omezení proměnných je počet těchto řešení konečný ale jejich počet je extrémně vysoký. Proto je tato metoda vhodná pouze pro malé problémy s omezeným počtem diskrétních proměnných. Postup je možno zobecnit na úlohu MIP tak, že ke každé kombinaci diskrétních proměnných je vyřešena úloha LP kde jsou diskrétní proměnné považovány za konstanty.

Příklad 1: Uvažujme následující lineární program:

[1] Z - celá čísla (kladná i záporná)

$$
\begin{array}{rrl}
\text{Maximalizuj } z = & -2x_1 & + \ x_2 \\
\text{vzhledem k:} & 9x_1 & - \ 3x_2 \geq 11 \\
& x_1 & + 2x_2 \leq 10 \\
& 2x_1 & - \ x_2 \leq 7 \\
& x_1, x_2 \geq 0, & x_1, x_2 \in Z
\end{array}
$$

Z obrázku 1 je patrné, že v úvahu připadá 10 přípustných řešení s tím že optimální je $x_1 = 2, x_2 = 2$ se $z = -2$. Takovéto pozorování je však možné pouze vzhledem k jednoduchosti řešeného příkladu. Pokud nemáme k dispozici odpovídající obrázek 1, můžeme přípustná řešení odhadnout z omezení. Z druhé omezující podmínky, kde jsou koeficienty u jednotlivých proměnných nezáporné, můžeme usoudit, že $0 \leq x_1 \leq 10$ a $0 \leq x_2 \leq 5$. Dále pokud $x_2 = 0$, z prvního omezení plyne že $x_1 \geq 2$. A obráceně, pokud $x_2 = 5$, z třetího omezení plyne $x_1 \leq 6$. Pokud sloučíme všechny podmínky dohromady, získáme že $2 \leq x_1 \leq 6$ a $0 \leq x_2 \leq 5$. Z toho plyne, že tato úloha nemá více jak 30 přípustných řešení. Pokud prohledáme takovouto oblast nalezneme 20 nepřípustných a 10 přípustných. Optimum je v bodě $\mathbf{x} = (2, 2)$.



Obrázek 1: Přípustná oblast z příkladu 1.

Tato metoda se příliš nepoužívá. Pokud ano, tak především pro úlohy tzv. binárního LP, kde diskrétní proměnné nabývají pouze hodnot $\{0, 1\}$. Potom je možné prohledávání realizovat jako binární strom, což umožňuje redukovat počet nepřípustných stavů.

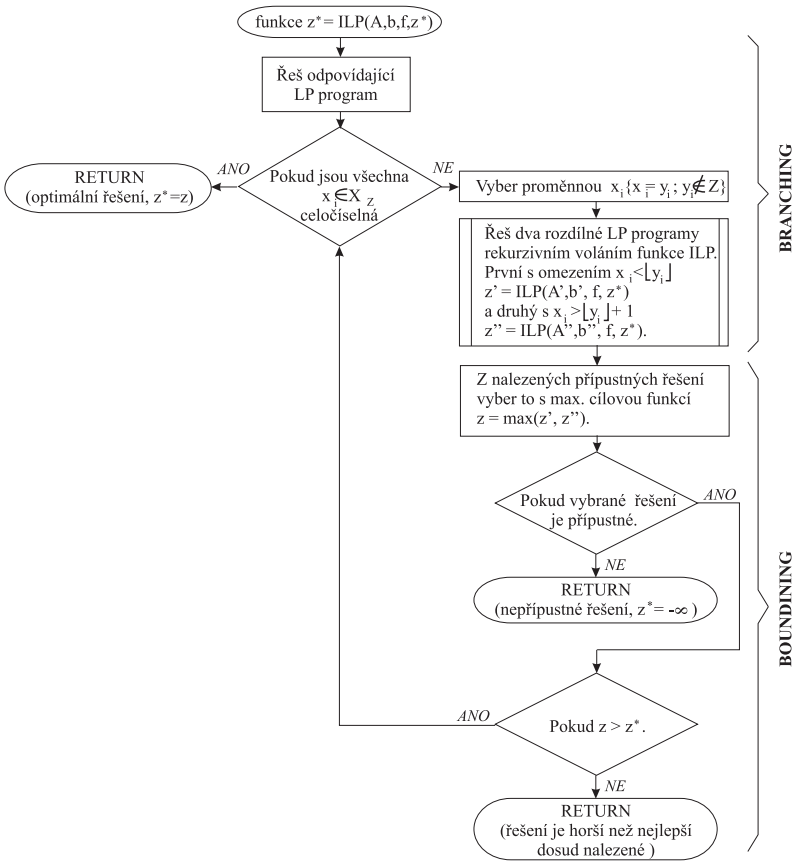
3 Metoda větví a mezí

Principem této metody je rozklad množiny přípustných řešení na disjunktní podmnožiny [1, 2, 4, 5]. Algoritmus začíná výpočtem úlohy tak, že zanedbává požadavek na celočíselnot a úloha je vyřešena klasickými metodami LP. Pokud jsou všechny proměnné $x_i \in X_Z$ celočíselné, potom výpočet končí. Pokud ne, vybere se libovolná proměnná $x_i = y_i, x_i \in X_Z$ taková že $y_i \notin Z$. Následně se vyšetřovaná oblast rozdělí na dvě podmnožiny tak že v první uvažujeme $x_i \leq \lfloor y_i \rfloor$ a v druhé $x_i \geq \lfloor y_i \rfloor + 1$. Výpočet LP je rekurzivně opakován pro obě nově vzniklé oblasti dokud není nalezeno přípustné řešení kde všechna $x_i \in X_Z$ jsou celočíselné.

Tímto způsobem algoritmus vytváří stavový prostor řešení který lze grafově znázornit stromem. Jeho vytváření se nazývá větvení (*branching*). Všechny uzly tohoto stromu obsahují nějaká částečná řešení problému. Listy (uzly které nemají následovníka) odpovídají buď nepřípustným řešením nebo nalezeným celočíselným řešením. Pokud algoritmus nalezne nějaké celočíselné řešení, může být hodnota odpovídající cílové funkce použita k prořezávání stromu (*bounding*) a tím k redukci počtu prohledávaných stavů. Tento mechanizmus pracuje tak, že pokud hodnota cílové funkce nějakého (i neceločíselného) řešení z je menší nebo rovna než z^* hodnota cílové funkce doposud nejlepšího nalezeného celočíselného řešení, tento uzel nevede k řešení s lepší hodnotou cílové funkce a potom může být tato větev odříznuta.

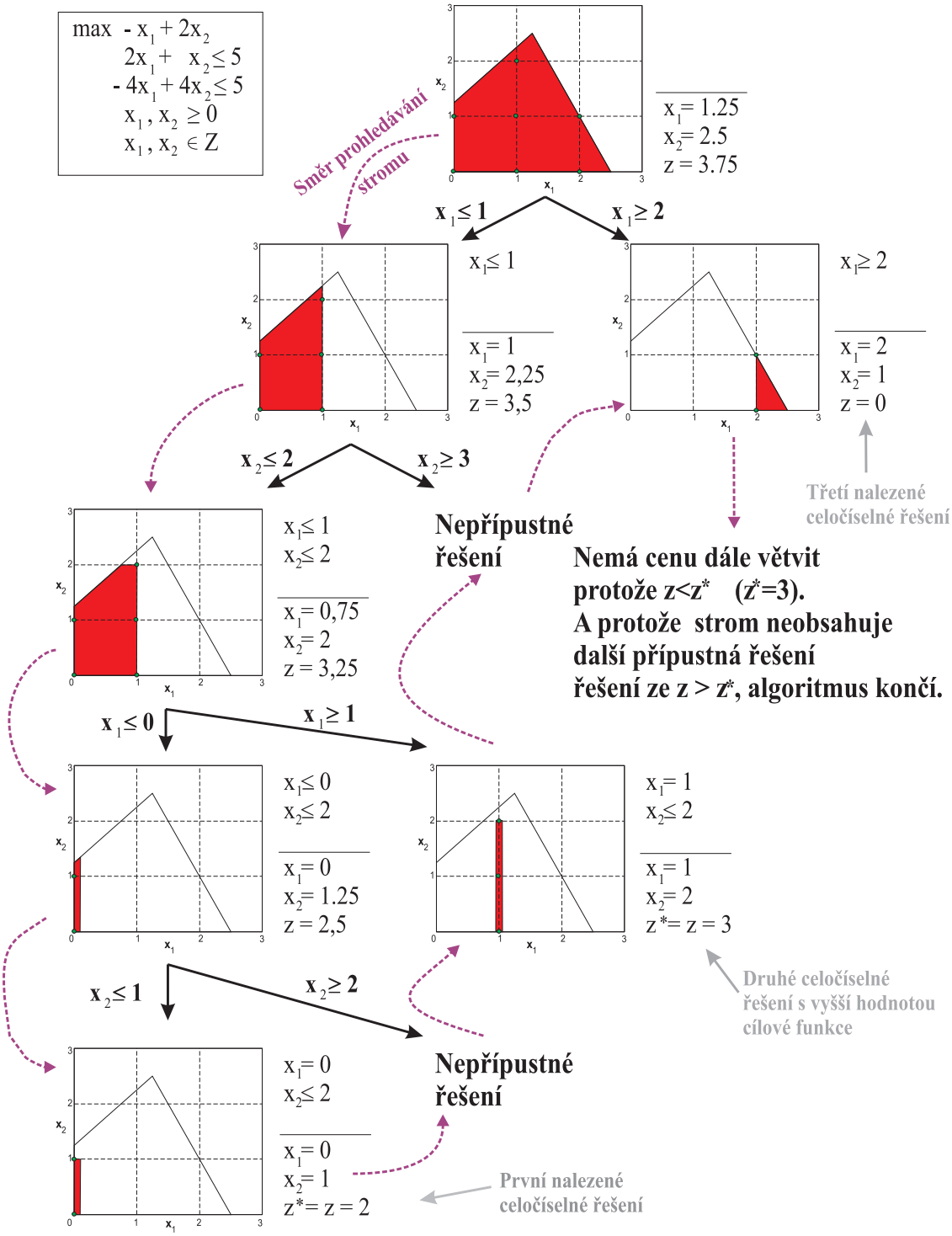
Algoritmus ILP nejčastěji využívá k řešení LP simplexovou metodu protože po přidání nové omezující podmínky není nutné spouštět simplexový algoritmus znova od začátku ale umožňuje navázat na předchozí výpočet LP řešením duální simplexové metody.

Celý algoritmus je znázorněn na obrázku 2. Dále je ukázán příklad ilustrující chování algoritmu. Průběh řešení příkladu 2 pomocí nástroje LP_SOLVE [6] je uveden v apendixu.



Obrázek 2: Algoritmus řešící úlohu ILP metodou větví a mezí.

Příklad 2:



Obrázek 3: Příklad 2 - metoda větví a mezí.

4 Metoda sečných nadrovin

Další skupinou algoritmů jsou metody sečných nadrovin (cutting plane method) [1, 2], založené podobně jako metoda větví a mezí na opakovaném řešení úlohy LP. Výpočet je prováděn iterativně, tak že v každém kroku je přidána další omezující podmínka zužující oblast přípustných řešení. Každá nová omezující podmínka musí splňovat tyto vlastnosti:

- Optimální řešení nalezené pomocí LP se stane nepřípustným.
- Žádné celočíselné řešení přípustné v předchozím kroku se nesmí stát nepřípustným.

Nové omezení splňující tyto vlastnosti je přidáno v každé iteraci. Vzniklý ILP program je vždy znovu řešen jako úloha LP. Proces je opakován, dokud není nalezeno přípustné celočíselné řešení. Konvergence takovéhoto algoritmu potom závisí na způsobu přidávání omezujících podmínek. Mezi nejznámější metody patří Dantzigovi řezy (*Dantzig cuts*) a Gomoryho řezy (*Gomory cuts*).

4.1 Dantzigovi řezy

Uvařujme úlohu celočíselného lineárního programování ve tvaru:

$$\begin{aligned} \text{Maximalizuj} \quad & \mathbf{cx} \\ \text{vzhledem k:} \quad & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{x} \geq 0 \text{ a } \mathbf{x} \in Z \end{aligned}$$

Dále uvažujme celočíselné prvky $\mathbf{A}$ a $\mathbf{b}$. Nejednodušší metodou sečných nadrovin je metoda *Dantzig cuts*. Úloha lineárního programování se skládá z takzvaných bázových $\mathbf{x_B}$ a nebázových proměnných $\mathbf{x_N}$ tj. proměnných reprezentující vlastní úlohu a proměnných reprezentující omezení typu $\leq$ nebo $\geq$. Potom můžeme soustavu omezení rozepsat

$$\begin{aligned} \mathbf{Ax} &= \mathbf{b} \\ [\mathbf{A_B}, \mathbf{A_N}] [\mathbf{x_B}, \mathbf{x_N}]^T &= \mathbf{b} \\ \mathbf{A_B x_B} + \mathbf{A_N x_N} &= \mathbf{b} \\ \mathbf{x_B} &= \mathbf{A_B}^{-1}\mathbf{b} - \mathbf{A_B}^{-1}\mathbf{A_N x_N} \end{aligned}$$

Pokud řešíme tuto úlohu pouze jako úlohu LP, potom $\mathbf{x_N}$ bude rovno nule. Potom optimální řešení získané LP bude

$$\mathbf{x_B} = \mathbf{A_B}^{-1}\mathbf{b} \tag{1}$$

Pokud je vektor $\mathbf{x_B}$ celočíselný, je toto řešení zároveň optimálním řešením úlohy ILP. V opačném případě, některé nebázové proměnné musí být větší než nula. Označme množinu nebázových proměnných Q. Protože nejmenší kladné celé číslo je jedna, musí platit

$$\sum_{x_j \in Q} x_j \geq 1 \tag{2}$$

Tím že přidáme toto omezení, původní optimální řešení se stane nepřípustným. Avšak nedojde k vyloučení žádného celočíselného řešení včetně optimálního. Celý proces je opakován, dokud není nalezeno první celočíselné řešení, které je zároveň optimální. Nevýhodou tohoto řešení je, že nezaručuje konvergenci v konečném počtu iterací. Konvergenci je možno zajistit úpravou vztahu (2) tak, že do omezení zahrneme jen některé proměnné

$$\sum_{x_j \in Q_i} x_j \geq 1, \; Q_i = \{j : j \in Q, a_{ij} \neq 0\}, \tag{3}$$

kde a_{ij} označuje prvek matice $\mathbf{A}$. Avšak i přes toto zlepšení, metoda není zdaleka tak efektivní jako Gomoryho řezy.

4.2 Gomoryho řezy

Druhá metoda se vyznačuje mnohem lepší schopností odřezávat přípustný prostor řešení. Uvažujme omezení ve tvaru

$$\sum_{j=1}^{n} a_j x_j = b. \tag{4}$$

Když omezíme obor hodnot a_j na množinu celých kladných čísel bude platit

$$\sum_{j=1}^{n} \lfloor a_j \rfloor x_j \leq b. \tag{5}$$

Kromě toho i levá strana této nerovnice musí být celočíselná

$$\sum_{j=1}^{n} \lfloor a_j \rfloor x_j \leq \lfloor b \rfloor . \tag{6}$$

Pokud použijeme novou proměnnou x_s vyjadřující "vůli" získáme vztah

$$\sum_{j=1}^{n} \lfloor a_j \rfloor x_j + x_s = \lfloor b \rfloor . \tag{7}$$

kde $x_s \geq 0$ a nabývá pouze celočíselných hodnot. Zavedením notace $a_j = \lfloor a_j \rfloor + f_j$, $b = \lfloor b \rfloor + f$, můžeme (4) přepsat na

$$\sum_{j=1}^{n} (\lfloor a_j \rfloor + f_j) x_j = \lfloor b \rfloor + f. \tag{8}$$

Odečtením vztahů (7) a (8) získáme

$$\sum_{j=1}^{n} f_j x_j - x_s = f. \tag{9}$$

Tento vztah se nazývá Gomoryho řezy. Aby LP řešení úlohy bez této nové funkce bylo přípustné, musela by proměnná x_s být záporná. To je však nepřípustné. Velkou výhodou je, že pokud je nová omezující podmínka přidána do simplexové tabulky, potom aktuální řešení stále vyhovují podmínkám optimality. Proto nejednodušším způsobem jak provést novou optimalizaci je použít duální simplexový algoritmus.

Algoritmus:

1. [Inicializace] Vyřeš úlohu jako úlohu LP.
2. [Test optimality] Pokud řešení je celočíselné, výpočet končí. V opačném případě proveď krok 3.
3. [Redukce] V simplexové tabulce vyber řádek r s největším $b_r > 0$. Pro tento řádek přidej odpovídající omezení (9) na konec tabulky. Přeoptimalizuj úlohu pomocí duálního LP (může potřeba i víc než jeden krok) a jdi na krok 2.

Příklad 3: Je zadána následující úloha ILP [7]:

$$\begin{array}{lcl} \text{Maximalizuj} & x_1 & + 2x_2 \\ \text{vzhledem k:} & -3x_1 & + 4x_2 \leq 6 \\ & 4x_1 & + 3x_2 \leq 12 \\ & x_1, x_2 \geq 0, & x_1, x_2 \in Z \end{array}$$

x_1	x_2	x_3	x_4	b
-3	4	1	0	6
4	3	0	1	12
-1	-2	0	0	0

x_1	x_2	x_3	x_4	b
-0,75	1	0,25	0	1,5
6,25	0	-0,75	1	7,5
-2,5	0	0,5	0	3

x_1	x_2	x_3	x_4	b
0	1	0,16	0,12	2,4
1	0	-0,12	0,16	1,2
0	0	0,2	0,4	6

Sestavíme simplexovaou tabulku, kterou vyřešíme bez ohledu na podmínku celočíselnosti.

Tím jsme získali neceločíselné optimální řešení. Ani jedna z proměnných není celočíselná, proto vybereme tu s největší desetinnou části, tj. x_1, tj. první řádek tabulky. Sestavíme novou podmínku (Gomoryho řez).

x_1	x_2	x_3	x_4	x_{s1}	b
0	1	0,16	0,12	0	2,4
1	0	-0,12	0,16	0	1,2
0	0	-0,16	-0,12	1	-0,4
0	0	0,2	0,4	0	6

Provedeme jedn krok duální simplexové metody.

x_1	x_2	x_3	x_4	x_{s1}	b
0	1	0	0	1	2
1	0	0	0,25	-0,75	1,5
0	0	1	0,75	-6,25	2,5
0	0	0	0,25	1,25	5,5

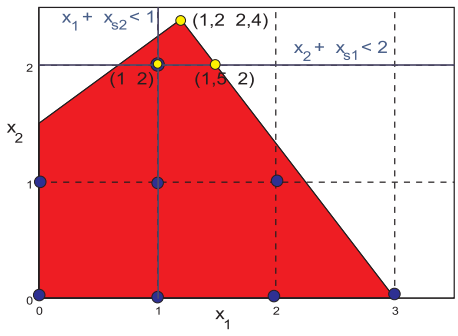
Protože x_1 není stále celočíselné, přidáme další omezující podmínku, kterou vytvoříme ze druhého řádku tabulky.

x_1	x_2	x_3	x_4	x_{s1}	x_{s2}	b
0	1	0	0	1	0	2
1	0	0	0,25	-0,75	0	1,5
0	0	1	0,75	-6,25	0	2,5
0	0	0	-0,25	0,75	1	0,5
0	0	0	0,25	1,25	0	5,5

Další tabulka je již optimální.

Při dodržení určitých pravidel si můžeme dovolit neudržovat v tabulce všechny řádky a sloupce. Gomoriho algoritmus dovoluje řešit i úlohy MIP. Nevýhodou proti metodě větví a mezí je, že dokud

x_1	x_2	x_3	x_4	b
0	1	0	0	2
1	0	0	0	1
0	0	0	0	5



Obrázek 4: Přípustná oblast z příkladu 3.

algoritmus nedoběhne, neví k dispozici žádný i třeba ne optimální výsledek. Metoda konverguje právě k jedinému celočíselnému optimálnímu řešení, bez toho aby nalezla jiná celočíselná řešení s horší hodnotou cílové funkce.

Metoda sečných nadrovin se dnes používá velice zřídka. Ve většině softwarových nástrojů se používá metody větví a mezí.

Reference

[1] A. Jensen and F. Bard, *Operations Research Models and Methods.* John Wiley & Sons, Inc., 2002.

[2] A. Eiselt and L. Sandblom, *Integer Programming and Network Models.* Springer Verlag, 2000.

[3] J. Demel, *Grafy a jejich aplikace.* Academia, second ed., 2002.

[4] R. Vanderbei, *Linear Programming : Foundations and Extensions.* http://www.princeton.edu/~rvdb/LPbook: Princeton University, second ed., 2001.

[5] F. S. Hiller and G. J. Lieberman, *Introduction to Operations Research.* McGraw-Hill, Inc, sith ed., 1995.

[6] J. C. Kantor, *LP_SOLVE 2.3.* ftp://ftp.es.ele.tue.nl/pub/lp_solve/, 1995.

[7] M. Berka, *Operační výzkum.* http://home.eunet.cz/berka/o/matempr1.htm.

[8] P. Klingerova, "Linární programování, diplomová práce," tech. rep., www.fm.vslib.cz/cgi-bin/toASCII/~ksi/cz/mater/oa/linprog/, 1997.

[9] J. Štecha, *Optimální rozhodování a řízení.* ČVUT, 2000.

A Řešení příkladu 2 pomocí programu LP_SOLVE

```
D:\Dev\ILP\LP40b\Release>lp40 -d -v6 ex1.lp
Model name:  lp
Objective:   Maximize(r_0)
Model size:      2 variables,     2 constraints,        6 non-zeros.
Variables:       2 integer,       0 semi-cont.,         0 SOS.
Constraints:     0 equality,      0 Lagrangean,         0 SOS.

 1--> starting milpsolve
 1--> A solution was found
 1--> x1                   1.25
 1--> x2                   2.5
Level 1 OPT     value 3.75
 1--> Unsatisfied truncated variables; Selecting var x1, val: 1.25
 1--> Current bounds:
 1--> starting floor subproblem with bounds:
 1--> x1 < 1
 2----> starting milpsolve
 2----> A solution was found
 2----> x1                   1
 2----> x2                   2.25
Level 2 OPT     value 3.5
 2----> Unsatisfied truncated variables; Selecting var x2, val: 2.25
 2----> Current bounds:
 2----> x1 < 1
 2----> starting floor subproblem with bounds:
 2----> x1 < 1
 2----> x2 < 2
 3------> starting milpsolve
 3------> A solution was found
 3------> x1                   0.75
 3------> x2                   2
Level 3 OPT     value 3.25
 3------> Unsatisfied truncated variables; Selecting var x1, val: 0.75
 3------> Current bounds:
 3------> x1 < 1
 3------> x2 < 2
 3------> starting floor subproblem with bounds:
 3------> x1 = 0
 3------> x2 < 2
 4--------> starting milpsolve
 4--------> A solution was found
 4--------> x1                   0
 4--------> x2                   1.25
Level 4 OPT     value 2.5
 4--------> Unsatisfied truncated variables; Selecting var x2, val: 1.25
 4--------> Current bounds:
 4--------> x1 = 0
 4--------> x2 < 2
 4--------> starting floor subproblem with bounds:
 4--------> x1 = 0
 4--------> x2 < 1
```

```
 5----------> starting milpsolve
 5----------> A solution was found
 5----------> x1                   0
 5----------> x2                   1
Level 5 OPT INT value 2
 5----------> --> valid solution found
*** new best solution: old: -1e+024, new: 2 ***
 4--------> starting ceiling subproblem with bounds:
 4--------> x1 = 0
 4--------> x2 = 2
 5----------> starting milpsolve
Level 5 INF
 3------> starting ceiling subproblem with bounds:
 3------> x1 = 1
 3------> x2 < 2
 4--------> starting milpsolve
 4--------> A solution was found
 4--------> x1                   1
 4--------> x2                   2
Level 4 OPT INT value 3
 4--------> --> valid solution found
*** new best solution: old: 2, new: 3 ***
 2----> starting ceiling subproblem with bounds:
 2----> x1 < 1
 2----> x2 > 3
 3------> starting milpsolve
Level 3 INF
 1--> starting ceiling subproblem with bounds:
 1--> x1 > 2
 2----> starting milpsolve
 2----> A solution was found
 2----> x1                   2
 2----> x2                   1
Level 2 OPT NOB value 0 bound 3
... but it was worse than the best so far; discarded!

Value of objective function: 3

Actual values of the variables:
x1                  1
x2                  2
```

Rozvrhování v systémech diskrétních událostí
cvičení č.1
Formulace úlohy v rozvrhování

Přemysl Šůcha (suchap@fel.cvut.cz)

3. října 2005

1 Aplikace integer lineárního programování

Celá řada praktických problémů týkajících se optimalizace (sem patří i rozvrhování) může být modelována a řešena pomocí integer (celočíselného[1] nebo také diskrétního) lineárního programování ILP (Integer Linear Programming). Tato úloha se od úlohy běžného lineárního programování LP liší vtom, že proměnné mohou nabývat jen celočíselných hodnot. Pokud některé proměnné mohou nabývat i neceločíselných hodnot, potom se úloha nazývá smíšené celočíselné programování MIP (Mixed Integer Programming) [1, 2].

1.1 Toky v síti

Definice 1.1 ***Tok a cirkulace.*** *Mějme orientovaný graf G.* Tokem v síti *nazýváme takové ohodnocení hran reálnými čísly $f : E(G) \to \mathbb{R}$, které pro každý vrchol v splňuje* ***Kirchhoffův zákon*** *[3].*

$$\sum_{e \in E^+(v)} f(e) = \sum_{e \in E^-(v)} f(e) \tag{1}$$

Kde $E^+(v)$ je množina hran s počátečním vrcholem v a $E^-(v)$ je množina hran s koncovým vrcholem v.

Poznámka: Tok od zdroje ke spotřebiči můžeme snadno převést na *cirkulaci* přidáním tzv. návratové hrany ze spotřebiče ke zdroji.

1.2 Lineární a celočíselné lineární programování

Úlohu lineárního programování lze matematicky formulovat jako úlohu nalezení extrému (maxima nebo minima) lineární funkce více proměnných při vedlejších podmínkách vyjádřených lineárními rovnicemi nebo nerovnostmi [4, 1, 2].

Definice 1.2 ***Obecná úloha lineárního programování*** *[5]. Nechť $b = (b_1, \ldots, b_m)'$ a $c = (c_1, \ldots, c_n)$ jsou dané vektory (m, n přirozená čísla a $n \geq 2$) a*

$$\mathbf{A} = \begin{pmatrix} a_{11} & \ldots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \ldots & a_{mn} \end{pmatrix} \tag{2}$$

daná číselná matice. Uvažujeme množinu M všech bodů $x = (x_1, \ldots, x_n)'$, které vyhovují systému rovnic:

[1] Z - celá čísla (kladná i záporná)

$$\mathbf{A}x = b \tag{3}$$

Potom úlohou ***lineárního programování*** *je najít v množině M aspoň jeden bod $\widetilde{x}$, pro který nabývá lineární funkce*

$$f(x) = c.x \tag{4}$$

extrémní hodnoty (tedy maxima nebo minima) na M.

Lineární funkci (4) nazýváme *cílovou funkcí* (objective function), množinu M množinou přípustných bodů příslušnou danému lineárnímu optimalizačnímu problému. Tato množina je určena soustavou lineárních omezení (linear constraints) v lineárním programování reprezentovanou maticí **A**.

Úlohy tohoto typu jsou řešitelné v polynomiálním čase (např. *metodou vnitřního bodu* - interior point method [4]). Připomeňme že *simplexovým algoritmus* je sice statisticky velmi rychlý ale obecně má exponenciální složitost [6]. Avšak pokud některé proměnné x_i omezíme tak, že smějí nabývat pouze celočíselných hodnot mluvíme o takzvané úloze *integer lineárního programování* ILP. Tyto úlohy již není možné řešit v polynomiálním čase. Při jejich řešení se často používají algoritmy založené na metodě větví a mezí (*Branch and Bound*) [1, 2, 7].

1.3 Celočíselné lineární programování a totálně unimodulární matice

Obecná úloha ILP není řešitelná v polynomiálním čase. Existují však speciální případy, které lze řešit v polynomiálním čase [6].

Definice 1.3 ***Totálně unimodulární matice****. Řekneme, že matice $\mathbf{A} = [a_{ij}]$ typu m/n je totálně unimodulární, jestliže*

1. *$a_{ij} \in \{0, 1, -1\}$*
2. *determinant každé čtvercové podmatice matice **A** je roven 0; 1 nebo -1.*

Věta 1.1 *Je-li **A** totálně unimodulární a vektor b je celočíselný, pak se optimální řešení úlohy ILP shoduje s řešením příslušné reálné úlohy LP.*

Věta 1.2 *Úlohu ILP s totálně unimodulární maticí **A** a celočíselným vektorem b lze řešit simplexovým algoritmem a výsledné řešení je celočíselné.*

Věta 1.3 *Úloha ILP s totálně unimodulární maticí **A** a celočíselným vektorem b je řešitelná v polynomiálním čase.*

Věta 1.4 *Necha **A** je matice typu m/n taková, že*

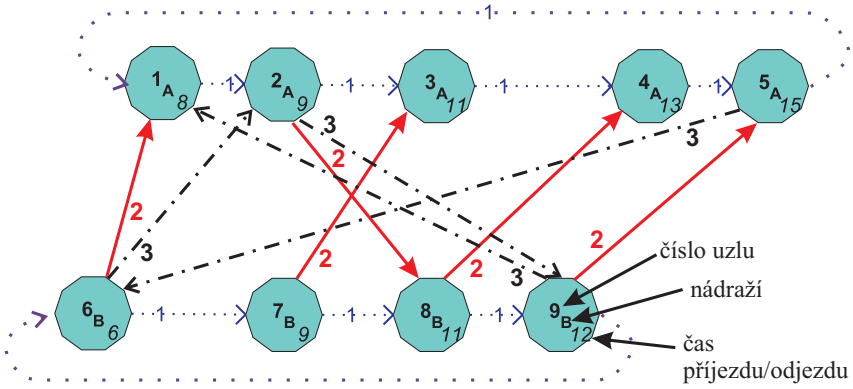
1. *$a_{ij} \in \{0, 1, -1\}$, $i = 1, ..., m$, $j = 1, ..., n$*
2. *každý sloupec matice **A** obsahuje buďto nejvýše jeden nenulový prvek nebo právě dva nenulové prvky, a to $+1$ a -1.*

*Pak je matice **A** totálně unimodulární.*

1.4 Plánování oběhu lokomotiv

Úkol: Je dána železniční síť a na ní jízdní řád zadaný grafem G na obr. 1. Vrcholy, které představují totéž nádraží v různých okamžicích, jsou nakresleny na vodorovné přímce. V grafu jsou tři druhy hran, které vyjadřují možnost čekání ve stanici (1), hrany, které vyjadřují převoz nákladu mezi stanicemi podle jízdního řádu (2). A konečné třetí typ hran, který reprezentuje přejezd lokomotivy bez nákladu z jednoho nádraží na druhé.

Uzly tedy reprezentují příjezdy a odjezdy vlaků na jednotlivých nádražích v daném čase. Hrany reprezentují čekání nebo přejezdy lokomotiv mezi dvěma nádražími. Zformulujte tento problém jako úlohu ILP. a) Minimalizujte náklady spojené s provozem lokomotiv (přejezdy lokomotiv a jejich prostoje na nádražích). Kolik lokomotiv je v tomto případě potřeba? b) Jak by jste modifikovali úlohu, když by jste chtěli minimalizovat počet potřebných lokomotiv.



Obrázek 1: Část grafu k hledání jízdního řádu.

Návod: Jedná se vlastně o úlohu hledání nejlevnější přípustné cirkulace v síti. K řešení použijte ILP. Proměnné x_i budou reprezentovat tok $f(e)$ hranami e grafu G. Řádky matice $\mathbf{A}$ budou vyjadřovat Kirchhoffův zákon (1) pro všechny uzly v grafu G (kolik lokomotiv do uzlu "přijede", musí z něho zase "odjet"). Označme hrany typu 1 jako množinu E_w, typu 2 jako množinu E_t a typu 3 jako množinu E_m, potom můžeme (1) rozepsat:

$$\forall v \sum_{e \in E_w^+(v)} f(e) - \sum_{e \in E_w^-(v)} f(e) + \sum_{e \in E_m^+(v)} f(e) - \sum_{e \in E_m^-(v)} f(e) + \sum_{e \in E_t^+(v)} f(e) - \sum_{e \in E_t^-(v)} f(e) = 0 \tag{5}$$

Tok $f(e)$ hranami $e \in E_w$ a $e \in E_m$ bude omezen zdola nulou a shora maximálním přípustným tokem $C(e)$. Hodnota $C(e)\ \forall\ e \in E_w$ reprezentuje kapacitu nádraží (počet lokomotiv které zde smějí čekat) a $C(e)\ \forall\ e \in E_m$ určuje kapacitu trati pro přejezd lokomotiv bez nákladu. Tok hranami $e \in E_t$ položme roven jedné $f(e) = 1\ \forall\ e \in E_t$. Tím docílíme, že právě jedna lokomotiva musí převést náklad z jednoho uzlu do druhého. Za předpokladu že z daného uzlu i může vyjet a přijet maximálně jedna lokomotiva, můžeme upravit soustavu omezujících podmínek (5) následovně:

$$\forall v \sum_{e \in E_w^+(v)} f(e) - \sum_{e \in E_w^-(v)} f(e) + \sum_{e \in E_m^+(v)} f(e) - \sum_{e \in E_m^-(v)} f(e) = -b_i \tag{6}$$

kde b_i je rovno 1 pokud se jedná o příjezd lokomotivy s nákladem na nádraží, -1 pokud se jedná o odjezd lokomotivy s nákladem z nádraží a 0 pokud žádná lokomotiva s nákladem nepřijíždí či neodjíždí a nebo se jedná o současný příjezd a odjezd.

Cílová funkce f je potom dána náklady na prostoje lokomotiv na jednotlivých nádražích $c_w(e)$ (v případě hran $e \in E_w$). A $c_m(e)$ náklady na přejezd lokomotiv bez nákladu(v případě hran $e \in E_m$). Náklady na převoz nákladu není nutné uvažovat protože jsou fixní a není možné je nikterak snížit. Formulace problému z obrázku obr. 1 v jazyce lineárního programování může vypadat následovně:

$$\underbrace{\left(\begin{array}{ccccc|cccc|cccc} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & -1 \end{array}\right)}_{\mathbf{A}} \cdot x = -\underbrace{\left(\begin{array}{c} 1 \\ -1 \\ 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ 0 \\ -1 \end{array}\right)}_{b} \tag{7}$$

$$min\left(\underbrace{\left(\begin{array}{ccccccccc|cccc} c_w & c_w & c_w & c_w & c_w & c_w & c_w & c_w & c_w & c_m & c_m & c_m & c_m \end{array}\right)}_{c} \cdot x\right) \tag{8}$$

Znovu si povšimněte, že jednotlivé řádky odpovídají uzlům grafu a sloupce reprezentují jednotlivé hrany.

Protože matice **A** v rovnici (7) je **totálně unimodální** a vektor b je **celočíselný**, je možné tuto úlohu ILP řešit simplexovým algoritmem jako úlohu LP v polynomiálním čase. K řešení použijte nástroj GLPK [8], který dokáže řešit úlohy LP, ILP a MIP. Nástroj je včleněn do Scheduling toolbox-u, který najdete na adrese http://dce.felk.cvut.cz/rdu/tajne/sch-0-1-2-1.zip. Řešení příkladu z obrázku 1 pomocí toolboxu naleznete v příkladu http://dce.felk.cvut.cz/rdu/rozvrhovani/download/rdu_c1.m.

Poznámka: Úlohu tohoto typu lze samozřejmě řešit také grafovými algoritmy (např. *Out of Kilter* [3]), který je také řešitelný také v polynomiálním čase.

Reference

[1] P. Klingerova, "Linární programování, diplomová práce," tech. rep., http://www.fm.vslib.cz/cgi-bin/toASCII/~ksi/cz/mater/oa/linprog/index.htm, 1997.

[2] R. Vanderbei, *Linear Programming : Foundations and Extensions*. http://www.princeton.edu/~rvdb/LPbook: Princeton University, second ed., 2001.

[3] J. Demel, *Grafy a jejich aplikace*. Academia, second ed., 2002.

[4] J. Štecha, "Optimální rozhodování a řízení," tech. rep., 2000.

[5] K. Rektorys and kol., *Přehled užité matematiky II*. Prometheus, sixth ed., 1995.

[6] Z. Ryjáček, "Teorie grafů a diskrétní optimalizace 2," tech. rep., cam.zcu.cz/~ryjacek/students/ps/TGD2.pdf, 2001.

[7] P. Šůcha, "Celočíselné lineární programování," tech. rep., http://dce.felk.cvut.cz/sucha/articles/ilp.pdf, 2004.

[8] A. Makhorin, *GLPK (GNU Linear Programming Kit) Version 4.6*. http://www.gnu.org/software/glpk/, 2004.

[9] J. C. Kantor, *LP_SOLVE 2.3*. ftp://ftp.es.ele.tue.nl/pub/lp_solve/, 1995.