# 5. Exercise Session

15.11.

- Prog. Exercise 1
- Simplex with upper bounds

## PA1: Revised Simplex Method

Overview: Solve LPs with the Rev. Simplex Method, using exact Fractions

(written in Java/C/C++)

Step 1: parse LP file (you can the provided, see Javadoc)
Step 2: find initial solution via Phase I (or Big M)
Step 3: find an optimal sol. via Phase II
Step 4: write down the opt. sol./obj. value to a file

---

Details: storing during calculation
- CARRY-Matrix,
- matrix $A$
- set of basis variables

CARRY

$$\begin{array}{|c|c|}\hline -z & -\pi \\ \hline \bar{b} & B^{-1} \\ \hline \end{array}$$

- Init: $\min c^T x$

$$Ax = b_1, \quad \geq b_2, \quad \leq b_3$$

- add slack/surplus var.

CARRY(0):

$$\begin{array}{|c|c|}\hline -z_0 & 0 \\ \hline b & I \\ \hline \end{array}$$

$\rightarrow \min c^T x'$, $A'x' = b$

- introduce $x^+/x^-$ for unbounded var. s.th. $x^+, x^- \geq 0$

$\rightarrow \min c^T x''$, $A''x'' = b$

- multiply row $i$, where $b_i < 0$ with $-1$, to obtain a non-neg. RHS

- store information about, surplus/slack/artif, $x^+$, $x^-$ variables ⇝ transform back after solving

**Phase I** • introduce artifial var. $x^a$

$$(I_n | A)\begin{pmatrix} x^a \\ x \end{pmatrix} = b, \quad \text{sol. is } x^a = b$$

- can use slack var. as art. variables

$$\begin{aligned} x_1 + x_2 &\le 2 \\ -x_1 + x_3 &= -3 \\ x_2 + x_3 &\ge 4 \end{aligned} \quad \rightsquigarrow \quad \begin{aligned} x_1 + x_2 \quad + s_1 &= 2 \\ x_1 \quad - x_3 \quad &= 3 \\ x_2 + x_3 - s_3 &= 4 \end{aligned}$$

⇓

$$\begin{aligned} x_1 + x_2 \quad + s_1 \quad &= 2 \\ x_1 \quad - x_3 \quad + a^1 &= 3 \\ x_2 + x_3 - s_3 + a^2 &= 4 \end{aligned}$$

⇝ sol: $s_1 = 2$
$a^1 = 3 \quad a^2 = 4$

introduce cost coeffs: $c_{x_a} = 1$, $c_x = 0$

**Parsing the LP:**

- given parser ⟶ read LP file, ask for obj. coeff, var, coeff,

answers are doubles: $0.333 x_1 + \ldots$

⇝ transform to fraction: $\frac{333}{1000}$

maybe transform input:

$\cancel{1/3\, x_1 + 2/3\, x_2 = 4}$

$x_1 + 2x_2 = 12$

steps in the simplex algorithm:

1. pricing: $\bar{c}_j := c_j - \pi' A_j$, one at a time until a neg. $\bar{c}_j$ is found

(no $\bar{c}_j < 0 \Rightarrow$ found an opt. sol)

But: consider other choices

find $j$, s.th. – $\theta^*$ is maximal

– obj. decreases the most

2. col. gen: $x_j$ as $x_j = B^{-1} A_j$

determ. $\theta^*$ (avoid cycling, Bland's rule)

3. pivot step: update CARRY

$CARRY^{(l)} \rightarrow CARRY^{(l+1)}$ by pivoting on $x_{lj}$

4. update the basis, $j$ leaves basis, $l$ enters basis

---

$CARRY^{(l)} \rightarrow CARRY^{(l+1)} = P_l \cdot \ldots \cdot P_1 \cdot CARRY^{(0)}$

Optional idea:

$P = (e_1, e_2, \ldots, e_{l-1}, \eta, e_{l+1}, \ldots, e_n)$

$e_i = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \leftarrow$ i-th pos.

$\eta = \begin{pmatrix} -\frac{x_{1j}}{x_{lj}} \\ -\frac{x_{2j}}{x_{lj}} \\ \vdots \end{pmatrix}$

$$\begin{array}{cl} \begin{pmatrix} \frac{1}{x_{lj}} \\ \vdots \\ -\frac{x_{m+1,j}}{x_{lj}} \end{pmatrix} & \begin{array}{l} \leftarrow l\text{-th row} \\ \\ \\ \end{array} \end{array}$$

→ store chain of $\eta$ vectors as pairs of $(j, l)$

## Numerical Stab.

- shouldn't be an issue with exact fractions
- use arb. precision denom., nominator types like BigInteger (C/C++: use gmp)

class 'Fraction' needs methods like:

+, −, *, /, < (less), ≤, to_double,

⇝ simplify fractions after operations

## Requirements

- reading LP files (CPLEX LP format)
- calc. w. exact frac.
- solving all LPs in general (afiro, kb2, sc50a/b in "short" time (minutes))
- giving obj/values in form of sol. file

Line 1: "sol. status: <status>"

2: "objective value: <value>"

3+: "<variable name> <value>"

only non zero var.

Example:

| | |
|---|---|
| solution status: | optimal |
| obj. value: | −1/3 |
| x1 | 1/10 |
| x2 | 2/3 |

sol. status: unbounded / infeas.

- command line opt.:
  java Simplex.java lp.lp -o solution.sol
  $\rightsquigarrow$ solves lp.lp, writes sol. to solution.sol
  (opt. add. parameter)
- GUI is optional / not needed

Hints & Advises:
- sol. in terms of orig. problem (not slack)
- no additional libs needed / allowed
- implement anticycling rule
- write code yourself

---

LPs with upper bounds (assume, lower bound = 0)

$\min c^T x$
$Ax = b$
$0 \le x \le u$

so far: $x$ is bfs

$x \to x_B$ corresponds to $B$
$x \to x_N = 0$

$A \to B$, $A \to N$
$\Rightarrow Ax = Bx_B + \cancel{N x_N}$
$\Rightarrow x_B = B^{-1} b$

now:

$x \to x_B$ corres. to $B$
$x \to x_L$ corres. to $L$
$x \to x_U$ corres. to $U$

$\Big\}$ $B, L, U$ is a partition of $A$

$A \to B$, $A \to L$, $A \to U$

$x_i = l_i$ for $x_i \in L$
$x_i = u_i$ for $x_i \in U$

$\Rightarrow Ax = Bx_B + \underbrace{\cancel{L \cdot x_L}}_{=0} + \underbrace{U \cdot x_u}_{\text{constand}}$

$$\boxed{x_B = B^{-1}b - B^{-1} U \cdot x_u}$$

<u>Thm:</u> LP with upper bounds has an opt. sol. $\Leftrightarrow$ there is an opt. bfs (form above)

<u>Proof:</u> LP with upper bounds can be rewritten in Stand. form

$$(P') := \begin{array}{l} \min c^T x \\ Ax = b \\ x + s = u \\ x \geq 0 \\ s \geq 0 \end{array}$$

we know: $P'$ has an opt sol.

$\Rightarrow P'$ has an opt. bfs with Basis $B'$

$B' x_B = \binom{b}{u}$

$$\begin{pmatrix} B & B_Q & 0 & 0 \\ I_p & 0 & I_q & 0 \\ 0 & I_q & 0 & 0 \\ 0 & 0 & 0 & I_s \end{pmatrix} \begin{pmatrix} x_j : x_j \in B', s_j \in B' \\ x_j : x_j \in B', s_j \notin B' \\ s_j : x_j \in B', s_j \in B' \\ s_j : x_j \notin B', s_j \in B' \end{pmatrix} = \begin{pmatrix} b \\ u \end{pmatrix} \begin{array}{l} \}\, m \\ \}\, n \end{array}$$

<u>row 1:</u> $Bx_B = b$ "$Ax = b$" $B$

<u>row 2:</u> $x_j \neq 0, s_j \neq 0 \Rightarrow x_j + s_j = u_j$

<u>row 3:</u> $x_j \neq 0,\ s_j = 0 \Rightarrow x_j = u_j$ $U$

<u>row 4:</u> $x_j = 0,\ s_j \neq 0 \Rightarrow s_j = u_j$ $L$

$\Rightarrow$ x var. from bfs are an opt. sol for the LP with upper bounds

- add. constr. (upper bounds) are fulfilled $\square$

A sol. $x$ for an LP with UB is optimal

$$\Updownarrow$$

$\bar{c}_j = 0 \qquad x_j \in B$

$\bar{c}_j \geq 0 \qquad x_j \in L \qquad$ ( $c^T x$ is increasing by incr. $x_j$)

$\bar{c}_j \leq 0 \qquad x_j \in U \qquad$ ( $c^T x$ is decreasing by incr. $x_j$)

$\rightsquigarrow$ Adjust pivot step in simplex algorithm