Linear Programming - Tableaus

Tableau

Consider a linear program in the standard form:

$$\begin{aligned} \text{minimize} \quad & c^\top x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

Tableau

Consider a linear program in the standard form:

$$\begin{aligned} \text{minimize} \quad & c^\top x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

We have considered the simplex algorithm, which searches for the minimum by moving around the vertices of the feasible region.

Tableau

Consider a linear program in the standard form:

$$\begin{aligned} \text{minimize} \quad & c^\top x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

We have considered the simplex algorithm, which searches for the minimum by moving around the vertices of the feasible region.

The algorithm is relatively straightforward but, in its original form, not so suitable for computations by hand.

Tableau

Consider a linear program in the standard form:

$$\begin{aligned} \text{minimize} \quad & c^\top x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

We have considered the simplex algorithm, which searches for the minimum by moving around the vertices of the feasible region.

The algorithm is relatively straightforward but, in its original form, not so suitable for computations by hand.

Tableaus provide all information about the current state of the simplex algorithm and can be used to streamline the process.

Keep in mind that we are not developing a new algorithm. Tableau just provides another view of the same simplex algorithm as presented before.

Tableau (Matrix Form)

Consider LP with a matrix A and vectors b, c. Assume $A = (B\ N)$ where B consists of basic columns and N of the non-basic ones.

Tableau (Matrix Form)

Consider LP with a matrix A and vectors b, c. Assume $A = (B\ N)$ where B consists of basic columns and N of the non-basic ones.

Consider the following matrix (the *initial tableau*):

$$\begin{pmatrix} A & b \\ c^\top & 0 \end{pmatrix} = \begin{pmatrix} B & N & b \\ c_B^\top & c_N^\top & 0 \end{pmatrix}$$

Tableau (Matrix Form)

Consider LP with a matrix A and vectors b, c. Assume $A = (B\ N)$ where B consists of basic columns and N of the non-basic ones.

Consider the following matrix (the *initial tableau*):

$$\begin{pmatrix} A & b \\ c^\top & 0 \end{pmatrix} = \begin{pmatrix} B & N & b \\ c_B^\top & c_N^\top & 0 \end{pmatrix}$$

Apply elementary row operations so that the matrix B is turned into I_m (preserving the last row for now). That is, multiply with

$$\begin{pmatrix} B^{-1} & 0 \\ 0 & 1 \end{pmatrix}$$

The result is

$$\begin{pmatrix} B^{-1} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} B & N & b \\ c_B^\top & c_N^\top & 0 \end{pmatrix} = \begin{pmatrix} I_m & B^{-1}N & B^{-1}b \\ c_B^\top & c_N^\top & 0 \end{pmatrix}$$

Tableau (Matrix Form)

We have

$$\begin{pmatrix} I_m & B^{-1}N & B^{-1}b \\ c_B^\top & c_N^\top & 0 \end{pmatrix}$$

Tableau (Matrix Form)

We have

$$\begin{pmatrix} I_m & B^{-1}N & B^{-1}b \\ c_B^\top & c_N^\top & 0 \end{pmatrix}$$

We apply row operations to the last row to eliminate the $c_B^\top$. This corresponds to multiplying the matrix with

$$\begin{pmatrix} I_m & 0 \\ -c_B^\top & 1 \end{pmatrix}$$

Tableau (Matrix Form)

We have

$$\begin{pmatrix} I_m & B^{-1}N & B^{-1}b \\ c_B^\top & c_N^\top & 0 \end{pmatrix}$$

We apply row operations to the last row to eliminate the $c_B^\top$. This corresponds to multiplying the matrix with

$$\begin{pmatrix} I_m & 0 \\ -c_B^\top & 1 \end{pmatrix}$$

We obtain

$$\begin{pmatrix} I_m & 0 \\ -c_B^\top & 1 \end{pmatrix} \begin{pmatrix} I_m & B^{-1}N & B^{-1}b \\ c_B^\top & c_N^\top & 0 \end{pmatrix} = \begin{pmatrix} I_m & B^{-1}N & B^{-1}b \\ 0 & c_N^\top - c_B^\top B^{-1}N & -c_B^\top B^{-1}b \end{pmatrix}$$

This is the *canonical form tableau for the basis B*.

Tableau (Components)

Let $A = (u_1 \ \ldots, u_n)$, the basis $\{x_1, \ldots, x_m\}$, $B = (u_1 \ \ldots, u_m)$.

Assume $u_k = (u_{1k}, \ldots, u_{nk})$. Then the initial tableau is

$$\begin{pmatrix} B & N & b \\ c_B^\top & c_N^\top & 0 \end{pmatrix} = \begin{pmatrix} u_{11} & \cdots & u_{1m} & u_{1(m+1)} & \cdots & u_{1n} & b_1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ u_{m1} & \cdots & u_{mm} & u_{m(m+1)} & \cdots & u_{mn} & b_m \\ c_1 & \cdots & c_m & c_{m+1} & \cdots & c_n & 0 \end{pmatrix}$$

Tableau (Components)

Let $A = (u_1 \ \ldots, u_n)$, the basis $\{x_1, \ldots, x_m\}$, $B = (u_1 \ \ldots, u_m)$.

Assume $u_k = (u_{1k}, \ldots, u_{nk})$. Then the initial tableau is

$$\begin{pmatrix} B & N & b \\ c_B^\top & c_N^\top & 0 \end{pmatrix} = \begin{pmatrix} u_{11} & \cdots & u_{1m} & u_{1(m+1)} & \cdots & u_{1n} & b_1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ u_{m1} & \cdots & u_{mm} & u_{m(m+1)} & \cdots & u_{mn} & b_m \\ c_1 & \cdots & c_m & c_{m+1} & \cdots & c_n & 0 \end{pmatrix}$$

Now transform all columns of the upper part of the matrix (except the last row) to the basis B:

$$u_k = B(y_{1k}, \ldots, y_{mk})^\top \text{ for } k = 1, \ldots, n \text{ and } b' = B^{-1}b$$

Tableau (Components)

Let $A = (u_1 \ldots, u_n)$, the basis $\{x_1, \ldots, x_m\}$, $B = (u_1 \ldots, u_m)$.

Assume $u_k = (u_{1k}, \ldots, u_{nk})$. Then the initial tableau is

$$\begin{pmatrix} B & N & b \\ c_B^\top & c_N^\top & 0 \end{pmatrix} = \begin{pmatrix} u_{11} & \cdots & u_{1m} & u_{1(m+1)} & \cdots & u_{1n} & b_1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ u_{m1} & \cdots & u_{mm} & u_{m(m+1)} & \cdots & u_{mn} & b_m \\ c_1 & \cdots & c_m & c_{m+1} & \cdots & c_n & 0 \end{pmatrix}$$

Now transform all columns of the upper part of the matrix (except the last row) to the basis B:

$$u_k = B(y_{1k}, \ldots, y_{mk})^\top \text{ for } k = 1, \ldots, n \text{ and } b' = B^{-1}b$$

and obtain $u_k = y_{1k}u_1 + \cdots + y_{mk}u_m$ for $k = m+1, \ldots, n$ and thus

$$\begin{pmatrix} 1 & \cdots & 0 & y_{1(m+1)} & \cdots & y_{1n} & b'_1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & y_{m(m+1)} & \cdots & y_{mn} & b'_m \\ c_1 & \cdots & c_m & c_{m+1} & \cdots & c_n & 0 \end{pmatrix}$$

Tableau (Components)

$$\begin{pmatrix} 1 & \cdots & 0 & y_{1(m+1)} & \cdots & y_{1n} & b'_1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & y_{m(m+1)} & \cdots & y_{mn} & b'_m \\ c_1 & \cdots & c_m & c_{m+1} & \cdots & c_n & 0 \end{pmatrix}$$

Tableau (Components)

$$\begin{pmatrix} 1 & \cdots & 0 & y_{1(m+1)} & \cdots & y_{1n} & b'_1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & y_{m(m+1)} & \cdots & y_{mn} & b'_m \\ c_1 & \cdots & c_m & c_{m+1} & \cdots & c_n & 0 \end{pmatrix}$$

Use row operations to eliminate $c_1, \ldots, c_m$. This is equivalent to multiplying the above matrix with

$$\begin{pmatrix} I_m & 0 \\ -c_B^\top & 1 \end{pmatrix} = \begin{pmatrix} 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & 0 \\ -c_1 & \cdots & -c_m & 1 \end{pmatrix}$$

from the left. We obtain ...

Tableau (Components)

... the canonical form for the basis $\{x_1, \ldots, x_m\}$:

$$\begin{pmatrix} 1 & \cdots & 0 & y_{1(m+1)} & \cdots & y_{1n} & b'_1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & y_{m(m+1)} & \cdots & y_{mn} & b'_m \\ 0 & \cdots & 0 & c'_{m+1} & \cdots & c'_n & -z \end{pmatrix}$$

Tableau (Components)

... the canonical form for the basis $\{x_1, \ldots, x_m\}$:

$$\begin{pmatrix} 1 & \cdots & 0 & y_{1(m+1)} & \cdots & y_{1n} & b'_1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & y_{m(m+1)} & \cdots & y_{mn} & b'_m \\ 0 & \cdots & 0 & c'_{m+1} & \cdots & c'_n & -z \end{pmatrix}$$

Here, $(b'_1, \ldots, b'_m)^\top = B^{-1}b$ is the vector b transformed to the basis B, and for $k = m+1, \ldots, n$ we have

$$c'_k = c_k - (y_{1k}c_1 + \cdots + y_{mk}c_m)$$

the reduced cost for the k-th column (non-basic).

Tableau (Components)

... the canonical form for the basis $\{x_1, \ldots, x_m\}$:

$$\begin{pmatrix} 1 & \cdots & 0 & y_{1(m+1)} & \cdots & y_{1n} & b'_1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & y_{m(m+1)} & \cdots & y_{mn} & b'_m \\ 0 & \cdots & 0 & c'_{m+1} & \cdots & c'_n & -z \end{pmatrix}$$

Here, $(b'_1, \ldots, b'_m)^\top = B^{-1}b$ is the vector b transformed to the basis B, and for $k = m+1, \ldots, n$ we have

$$c'_k = c_k - (y_{1k}c_1 + \cdots + y_{mk}c_m)$$

the reduced cost for the k-th column (non-basic). Also, note that the basic solution is $x = (b'_1, \ldots, b'_m, 0, \ldots, 0)$, and hence

$$-z = (-c_1)b'_1 + \cdots + (-c_m)b'_m$$

is the negative of the value of the objective for the basic solution corresponding to the basis $\{x_1, \ldots, x_m\}$.

Recall that, by definition, the basic solution x satisfies $x_{m+1} = \cdots = x_n = 0$.

Tableau Simplex

Assume that for a basis B we have obtained the canonical tableau:

$$\begin{pmatrix} 1 & \cdots & 0 & y_{1(m+1)} & \cdots & y_{1n} & b'_1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & y_{m(m+1)} & \cdots & y_{mn} & b'_m \\ 0 & \cdots & 0 & c'_{m+1} & \cdots & c'_n & -z \end{pmatrix}$$

The simplex algorithm then proceeds as follows:

1. Choose $i \in \{m+1, \ldots, n\}$ such that $c'_i < 0$.
2. Choose $j \in \{1, \ldots, m\}$ minimizing b'_j / y_{ji} over all j satisfying $y_{ji} > 0$.

   Note that $b'_j = x_j$ for the basic solution x w.r.t. B.
3. Move the i-the column into the basis and the j-th column out of the basis.
4. Use elementary row operations to transform the tableau into the canonical form for the new basis.
5. Repeat until $b'_1, \ldots, b'_m \geq 0$,

Example

$$\begin{aligned} x_1 + x_2 &\le 2 \\ x_1 &\le 1 \\ x_1, x_2 &\ge 0 \end{aligned}$$

Add slack variables x_3, x_4:

$$\begin{aligned} x_1 + x_2 + x_3 &= 2 \\ x_1 + x_4 &= 1 \\ x_1, x_2, x_3, x_4 &\ge 0 \end{aligned}$$

$$A = (u_1 \; u_2 \; u_3 \; u_4) = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

$x = (x_1, x_2, x_3, x_4)^\top$

$b = (2, 1)^\top$

$Ax = b$ where $x \ge 0$

$c = (-3, -2, 0, 0)^\top$

Example

$$\begin{aligned} x_1 + x_2 &\le 2 \\ x_1 &\le 1 \\ x_1, x_2 &\ge 0 \end{aligned}$$

Add slack variables x_3, x_4:

$$\begin{aligned} x_1 + x_2 + x_3 &= 2 \\ x_1 + x_4 &= 1 \\ x_1, x_2, x_3, x_4 &\ge 0 \end{aligned}$$

$$A = (u_1\; u_2\; u_3\; u_4) = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

$$x = (x_1, x_2, x_3, x_4)^\top$$

$$b = (2, 1)^\top$$

$$Ax = b \text{ where } x \ge 0$$

$$c = (-3, -2, 0, 0)^\top$$

Tableau for the basis $\{x_3, x_4\}$:

$$\left[\begin{array}{c|cccc|c} x_3 & 1 & 1 & 1 & 0 & 2 \\ x_4 & 1 & 0 & 0 & 1 & 1 \\ \hline -z & -3 & -2 & 0 & 0 & 0 \end{array}\right]$$

is already in the canonical form.

Note that the last row of the tableau corresponds to writing the objective as $-z + c^\top x = 0$ where z is a new variable and x is the basic solution for $\{x_3, x_4\}$.

Start with the basis $\{x_3, x_4\}$ and consider the canonical form:

$$\left[\begin{array}{c|cccc|c} x_3 & y_{31} & y_{32} & 1 & 0 & b_1 \\ x_4 & y_{41} & y_{42} & 0 & 1 & b_2 \\ \hline -z & c_1 & c_2 & c_3 & c_4 & 0 \end{array}\right] = \left[\begin{array}{c|cccc|c} x_3 & 1 & 1 & 1 & 0 & 2 \\ x_4 & 1 & 0 & 0 & 1 & 1 \\ \hline -z & -3 & -2 & 0 & 0 & 0 \end{array}\right]$$

Start with the basis $\{x_3, x_4\}$ and consider the canonical form:

$$\left[\begin{array}{c|cccc|c} x_3 & y_{31} & y_{32} & 1 & 0 & b_1 \\ x_4 & y_{41} & y_{42} & 0 & 1 & b_2 \\ \hline -z & c_1 & c_2 & c_3 & c_4 & 0 \end{array}\right] = \left[\begin{array}{c|cccc|c} x_3 & 1 & 1 & 1 & 0 & 2 \\ x_4 & 1 & 0 & 0 & 1 & 1 \\ \hline -z & -3 & -2 & 0 & 0 & 0 \end{array}\right]$$

Choose x_1 to enter the basis (x_1 has the reduced cost -3 and x_2 has the reduced costs -2).

Start with the basis $\{x_3, x_4\}$ and consider the canonical form:

$$\left[\begin{array}{c|cccc|c} x_3 & y_{31} & y_{32} & 1 & 0 & b_1 \\ x_4 & y_{41} & y_{42} & 0 & 1 & b_2 \\ \hline -z & c_1 & c_2 & c_3 & c_4 & 0 \end{array}\right] = \left[\begin{array}{c|cccc|c} x_3 & 1 & 1 & 1 & 0 & 2 \\ x_4 & 1 & 0 & 0 & 1 & 1 \\ \hline -z & -3 & -2 & 0 & 0 & 0 \end{array}\right]$$

Choose x_1 to enter the basis (x_1 has the reduced cost -3 and x_2 has the reduced costs -2).Now $b_1/y_{31} = 2/1 > 1/1 = b_2/y_{41}$. Thus, remove x_4 from the basis.

Start with the basis $\{x_3, x_4\}$ and consider the canonical form:

$$\left[\begin{array}{c|cccc|c} x_3 & y_{31} & y_{32} & 1 & 0 & b_1 \\ x_4 & y_{41} & y_{42} & 0 & 1 & b_2 \\ \hline -z & c_1 & c_2 & c_3 & c_4 & 0 \end{array}\right] = \left[\begin{array}{c|cccc|c} x_3 & 1 & 1 & 1 & 0 & 2 \\ x_4 & 1 & 0 & 0 & 1 & 1 \\ \hline -z & -3 & -2 & 0 & 0 & 0 \end{array}\right]$$

Choose x_1 to enter the basis (x_1 has the reduced cost -3 and x_2 has the reduced costs -2).Now $b_1/y_{31} = 2/1 > 1/1 = b_2/y_{41}$. Thus, remove x_4 from the basis.We move to the basis $\{x_1, x_3\}$ and transform the tableau into the canonical form for this basis:

$$\left[\begin{array}{c|cccc|c} x_2 & 1 & y_{12} & 0 & y_{14} & b_1' \\ x_4 & 0 & y_{32} & 1 & y_{34} & b_2' \\ \hline -z & c_1' & c_2' & c_3' & c_4' & 3 \end{array}\right] = \left[\begin{array}{c|cccc|c} x_1 & 1 & 0 & 0 & 1 & 1 \\ x_3 & 0 & 1 & 1 & -1 & 1 \\ \hline -z & 0 & -2 & 0 & 3 & 3 \end{array}\right]$$

Start with the basis $\{x_3, x_4\}$ and consider the canonical form:

$$\left[\begin{array}{c|cccc|c} x_3 & y_{31} & y_{32} & 1 & 0 & b_1 \\ x_4 & y_{41} & y_{42} & 0 & 1 & b_2 \\ \hline -z & c_1 & c_2 & c_3 & c_4 & 0 \end{array}\right] = \left[\begin{array}{c|cccc|c} x_3 & 1 & 1 & 1 & 0 & 2 \\ x_4 & 1 & 0 & 0 & 1 & 1 \\ \hline -z & -3 & -2 & 0 & 0 & 0 \end{array}\right]$$

Choose x_1 to enter the basis (x_1 has the reduced cost -3 and x_2 has the reduced costs -2).Now $b_1/y_{31} = 2/1 > 1/1 = b_2/y_{41}$. Thus, remove x_4 from the basis.We move to the basis $\{x_1, x_3\}$ and transform the tableau into the canonical form for this basis:

$$\left[\begin{array}{c|cccc|c} x_2 & 1 & y_{12} & 0 & y_{14} & b'_1 \\ x_4 & 0 & y_{32} & 1 & y_{34} & b'_2 \\ \hline -z & c'_1 & c'_2 & c'_3 & c'_4 & 3 \end{array}\right] = \left[\begin{array}{c|cccc|c} x_1 & 1 & 0 & 0 & 1 & 1 \\ x_3 & 0 & 1 & 1 & -1 & 1 \\ \hline -z & 0 & -2 & 0 & 3 & 3 \end{array}\right]$$

Here, the reduced cost of x_2 is -2, and of x_4 is 3. Thus, x_2 enters the basis.

Start with the basis $\{x_3, x_4\}$ and consider the canonical form:

$$\left[\begin{array}{c|cccc|c} x_3 & y_{31} & y_{32} & 1 & 0 & b_1 \\ x_4 & y_{41} & y_{42} & 0 & 1 & b_2 \\ \hline -z & c_1 & c_2 & c_3 & c_4 & 0 \end{array}\right] = \left[\begin{array}{c|cccc|c} x_3 & 1 & 1 & 1 & 0 & 2 \\ x_4 & 1 & 0 & 0 & 1 & 1 \\ \hline -z & -3 & -2 & 0 & 0 & 0 \end{array}\right]$$

Choose x_1 to enter the basis (x_1 has the reduced cost -3 and x_2 has the reduced costs -2).Now $b_1/y_{31} = 2/1 > 1/1 = b_2/y_{41}$. Thus, remove x_4 from the basis.We move to the basis $\{x_1, x_3\}$ and transform the tableau into the canonical form for this basis:

$$\left[\begin{array}{c|cccc|c} x_2 & 1 & y_{12} & 0 & y_{14} & b'_1 \\ x_4 & 0 & y_{32} & 1 & y_{34} & b'_2 \\ \hline -z & c'_1 & c'_2 & c'_3 & c'_4 & 3 \end{array}\right] = \left[\begin{array}{c|cccc|c} x_1 & 1 & 0 & 0 & 1 & 1 \\ x_3 & 0 & 1 & 1 & -1 & 1 \\ \hline -z & 0 & -2 & 0 & 3 & 3 \end{array}\right]$$

Here, the reduced cost of x_2 is -2, and of x_4 is 3. Thus, x_2 enters the basis.Now x_3 leaves the basis because $y_{12} > 0$ but $y_{32} = 0$.

Start with the basis $\{x_3, x_4\}$ and consider the canonical form:

$$\left[\begin{array}{c|cccc|c} x_3 & y_{31} & y_{32} & 1 & 0 & b_1 \\ x_4 & y_{41} & y_{42} & 0 & 1 & b_2 \\ \hline -z & c_1 & c_2 & c_3 & c_4 & 0 \end{array}\right] = \left[\begin{array}{c|cccc|c} x_3 & 1 & 1 & 1 & 0 & 2 \\ x_4 & 1 & 0 & 0 & 1 & 1 \\ \hline -z & -3 & -2 & 0 & 0 & 0 \end{array}\right]$$

Choose x_1 to enter the basis (x_1 has the reduced cost -3 and x_2 has the reduced costs -2).Now $b_1/y_{31} = 2/1 > 1/1 = b_2/y_{41}$. Thus, remove x_4 from the basis.We move to the basis $\{x_1, x_3\}$ and transform the tableau into the canonical form for this basis:

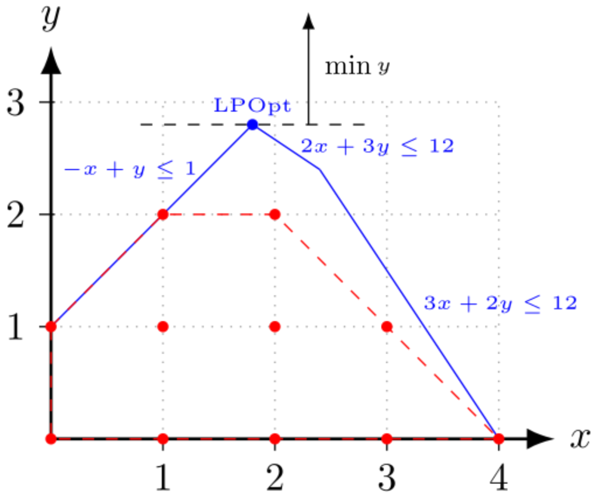
$$\left[\begin{array}{c|cccc|c} x_2 & 1 & y_{12} & 0 & y_{14} & b'_1 \\ x_4 & 0 & y_{32} & 1 & y_{34} & b'_2 \\ \hline -z & c'_1 & c'_2 & c'_3 & c'_4 & 3 \end{array}\right] = \left[\begin{array}{c|cccc|c} x_1 & 1 & 0 & 0 & 1 & 1 \\ x_3 & 0 & 1 & 1 & -1 & 1 \\ \hline -z & 0 & -2 & 0 & 3 & 3 \end{array}\right]$$

Here, the reduced cost of x_2 is -2, and of x_4 is 3. Thus, x_2 enters the basis.Now x_3 leaves the basis because $y_{12} > 0$ but $y_{32} = 0$.We move to the basis $\{x_1, x_2\}$ and transform the tableau into the canonical form:

$$\left[\begin{array}{c|cccc|c} x_1 & 1 & 0 & 0 & 1 & 1 \\ x_2 & 0 & 1 & 1 & -1 & 1 \\ \hline -z & 0 & 0 & 2 & 1 & 5 \end{array}\right]$$

Integer Linear Programming

Integer Linear Programming



ILP = LP + variables constrained to integer values

Integer Linear Programming

We consider several variants of integer programming:

- 0-1 integer linear programming
- Mixed 0-1 integer linear programming
- Integer linear programming
- Mixed integer linear programming

Integer Linear Programming

We consider several variants of integer programming:

- 0-1 integer linear programming
- Mixed 0-1 integer linear programming
- Integer linear programming
- Mixed integer linear programming

We consider the basic branch and bound algorithm.

Integer Linear Programming

We consider several variants of integer programming:

- 0-1 integer linear programming
- Mixed 0-1 integer linear programming
- Integer linear programming
- Mixed integer linear programming

We consider the basic branch and bound algorithm.

We also consider a cutting-plane method for integer programming.

Integer Linear Programming

We consider several variants of integer programming:

- 0-1 integer linear programming
- Mixed 0-1 integer linear programming
- Integer linear programming
- Mixed integer linear programming

We consider the basic branch and bound algorithm.

We also consider a cutting-plane method for integer programming.

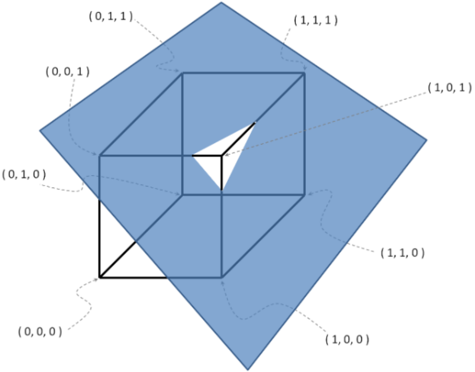
Integer linear programming is a huge subject; we shall only scratch its surface slightly.

0-1 Integer Linear Programming

Let us start with a special case where variables are constrained to values from $\{0, 1\}$.

0-1 integer linear program (0-1 ILP) is

$$\begin{aligned} \text{minimize} \quad & c^\top x \\ \text{subject to} \quad & Ax \leq b \\ & x_i \in \{0, 1\} \end{aligned}$$



0-1 Integer Linear Programming

Consider the following example:

$$\begin{aligned} \text{minimize} \quad & c^\top x \\ \text{subject to} \quad & a^\top x \leq b \\ & x \geq 0 \\ & x_i \in \{0, 1\} \end{aligned}$$

Here $c, a \in \mathbb{R}^n$ and $b \in \mathbb{R}$.

Do you recognize the problem?

0-1 Integer Linear Programming

Consider the following example:

$$\begin{aligned} \text{minimize} \quad & c^\top x \\ \text{subject to} \quad & a^\top x \leq b \\ & x \geq 0 \\ & x_i \in \{0, 1\} \end{aligned}$$

Here $c, a \in \mathbb{R}^n$ and $b \in \mathbb{R}$.

Do you recognize the problem? It is the 0-1 knapsack problem.

0-1 Integer Linear Programming

Consider the following example:

$$\begin{aligned} \text{minimize} \quad & c^\top x \\ \text{subject to} \quad & a^\top x \leq b \\ & x \geq 0 \\ & x_i \in \{0,1\} \end{aligned}$$

Here $c, a \in \mathbb{R}^n$ and $b \in \mathbb{R}$.

Do you recognize the problem? It is the 0-1 knapsack problem.

Theorem 1

Finding $x \in \{0,1\}^n$ satisfying the constraints of a given 0-1 integer linear program is NP-complete.

It is one of Karp's 21 NP-complete problems.

0-1 Mixed Integer Linear Programming

0-1 mixed integer linear program (0-1 MILP) is

$$\begin{aligned} \text{minimize} \quad & c^\top x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \\ & x_i \in \{0, 1\} \text{ for } x_i \in \mathcal{D} \end{aligned}$$

Here $\mathcal{D} \subseteq \{x_1, \ldots, x_n\}$ is a set of *binary variables*.

0-1 Mixed Integer Linear Programming

0-1 mixed integer linear program (0-1 MILP) is

$$\begin{array}{rl} \text{minimize} & c^\top x \\ \text{subject to} & Ax = b \\ & x \geq 0 \\ & x_i \in \{0, 1\} \text{ for } x_i \in \mathcal{D} \end{array}$$

Here $\mathcal{D} \subseteq \{x_1, \ldots, x_n\}$ is a set of *binary variables*.

The problem is NP-hard; the simplex algorithm cannot be used directly.

0-1 Mixed Integer Linear Programming

0-1 mixed integer linear program (0-1 MILP) is

$$\begin{array}{rl} \text{minimize} & c^\top x \\ \text{subject to} & Ax = b \\ & x \geq 0 \\ & x_i \in \{0,1\} \text{ for } x_i \in \mathcal{D} \end{array}$$

Here $\mathcal{D} \subseteq \{x_1, \ldots, x_n\}$ is a set of *binary variables*.

The problem is NP-hard; the simplex algorithm cannot be used directly.

The problem can be solved by searching for possible values 0 and 1 in the binary variables and solving the linear programs with binary variables fixed to concrete values.

0-1 Mixed Integer Linear Programming

0-1 mixed integer linear program (0-1 MILP) is

$$\begin{aligned} \text{minimize} \quad & c^\top x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \\ & x_i \in \{0, 1\} \text{ for } x_i \in \mathcal{D} \end{aligned}$$

Here $\mathcal{D} \subseteq \{x_1, \ldots, x_n\}$ is a set of *binary variables*.

The problem is NP-hard; the simplex algorithm cannot be used directly.

The problem can be solved by searching for possible values 0 and 1 in the binary variables and solving the linear programs with binary variables fixed to concrete values.

An exhaustive search through all possible binary assignments would be infeasible for many variables.

Usually, a sequential search that fixes only some of the binary variables and leaves the rest unrestricted to 0 or 1 is used.

Notation

In what follows, *LP relaxation* is the linear program obtained from 0-1 MILP by removing the constraints $x_i \in \{0, 1\}$ for $x_i \in \mathcal{D}$ and adding constraints $x_i \geq 0$ and $x \leq 1$ for all $x_i \in \mathcal{D}$.

Notation

In what follows, *LP relaxation* is the linear program obtained from 0-1 MILP by removing the constraints $x_i \in \{0, 1\}$ for $x_i \in \mathcal{D}$ and adding constraints $x_i \geq 0$ and $x \leq 1$ for all $x_i \in \mathcal{D}$.

Assume a global variable x^*, keeping the best solution satisfying the 0-1 MILP constraints. Initialized with the undefined symbol $\perp$.

Notation

In what follows, *LP relaxation* is the linear program obtained from 0-1 MILP by removing the constraints $x_i \in \{0, 1\}$ for $x_i \in \mathcal{D}$ and adding constraints $x_i \geq 0$ and $x \leq 1$ for all $x_i \in \mathcal{D}$.

Assume a global variable x^*, keeping the best solution satisfying the 0-1 MILP constraints. Initialized with the undefined symbol $\perp$.

Assume a global variable f^*, keeping the value of the best solution satisfying the 0-1 MILP constraints. Initialize with $f^* = \infty$.

Notation

In what follows, *LP relaxation* is the linear program obtained from 0-1 MILP by removing the constraints $x_i \in \{0, 1\}$ for $x_i \in \mathcal{D}$ and adding constraints $x_i \geq 0$ and $x \leq 1$ for all $x_i \in \mathcal{D}$.

Assume a global variable x^*, keeping the best solution satisfying the 0-1 MILP constraints. Initialized with the undefined symbol $\perp$.

Assume a global variable f^*, keeping the value of the best solution satisfying the 0-1 MILP constraints. Initialize with $f^* = \infty$.

Keep a pool of 0-1 MILP problems $\mathcal{P}$ initialized with $\mathcal{P} = \{P\}$ where P is the original 0-1 MILP to be solved.

Algorithm 1 Branch and Bound (Non-Deterministic)

repeat

Choose $P \in \mathcal{P}$

if LP relaxation of P is feasible **then**

Find a solution x of the LP relaxation of P

if $c^\top x < f^*$ **then**

if $x_i \in \{0, 1\}$ for all $x_i \in \mathcal{D}$ **then**

$x^* \leftarrow x$

$f^* \leftarrow c^\top x$

else

Choose $x_i \in \mathcal{D}$ such that $x_i \notin \{0, 1\}$

Generate LP P_0 by adding $x_i = 0$ to P

Generate LP P_1 by adding $x_i = 1$ to P

Add P_0 and P_1 to $\mathcal{P}$.

end if

end if

end if

$\mathcal{P} \leftarrow \mathcal{P} \setminus \{P\}$

until $\mathcal{P} = \emptyset$

Strategies

There are many possible strategies for choosing the problem to be solved next:

- DFS, BFS, etc.
- heuristics using solutions to the relaxations

Strategies

There are many possible strategies for choosing the problem to be solved next:

- DFS, BFS, etc.
- heuristics using solutions to the relaxations

There are heuristics for choosing the variable to be bounded:

- Simplest one: Choose x_i which maximizes $\min\{x_i, 1 - x_i\}$
- Look ahead to the relaxations of the possible subdivisions

Strategies

There are many possible strategies for choosing the problem to be solved next:

- DFS, BFS, etc.
- heuristics using solutions to the relaxations

There are heuristics for choosing the variable to be bounded:

- Simplest one: Choose x_i which maximizes $\min\{x_i, 1 - x_i\}$
- Look ahead to the relaxations of the possible subdivisions

The solutions to the LP relaxations can be reused. Some methods (dual simplex) exploit that we are just adding a single constraint $x_i = 0$ or $x_i = 1$.

Strategies

There are many possible strategies for choosing the problem to be solved next:

- DFS, BFS, etc.
- heuristics using solutions to the relaxations

There are heuristics for choosing the variable to be bounded:

- Simplest one: Choose x_i which maximizes $\min\{x_i, 1 - x_i\}$
- Look ahead to the relaxations of the possible subdivisions

The solutions to the LP relaxations can be reused. Some methods (dual simplex) exploit that we are just adding a single constraint $x_i = 0$ or $x_i = 1$.

The procedure may be stopped when we find a solution x, which gives a small enough value of the objective.

(Mixed) Integer Programming

Integer linear program (ILP) is

$$\begin{aligned} \text{minimize} \quad & c^\top x \\ \text{subject to} \quad & Ax \leq b \\ & x \geq 0 \\ & x \in \mathbb{Z}^n \end{aligned}$$

(Mixed) Integer Programming

Integer linear program (ILP) is

$$\begin{aligned} \text{minimize} \quad & c^\top x \\ \text{subject to} \quad & Ax \leq b \\ & x \geq 0 \\ & x \in \mathbb{Z}^n \end{aligned}$$

Mixed integer linear program (MILP) is

$$\begin{aligned} \text{minimize} \quad & c^\top x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \\ & x_i \in \mathbb{Z} \text{ for } x_i \in \mathcal{D} \end{aligned}$$

Here $\mathcal{D} \subseteq \{x_1, \ldots, x_n\}$ is a set of integer variables.

(Mixed) Integer Programming

Integer linear program (ILP) is

$$\begin{aligned} \text{minimize} \quad & c^\top x \\ \text{subject to} \quad & Ax \leq b \\ & x \geq 0 \\ & x \in \mathbb{Z}^n \end{aligned}$$

Mixed integer linear program (MILP) is

$$\begin{aligned} \text{minimize} \quad & c^\top x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \\ & x_i \in \mathbb{Z} \text{ for } x_i \in \mathcal{D} \end{aligned}$$

Here $\mathcal{D} \subseteq \{x_1, \ldots, x_n\}$ is a set of integer variables.

We may use a similar branch and bound approach as for the binary variables. The problem is that now, each integer variable has an infinite domain.

Notation

In what follows, *LP relaxation* is the linear program obtained from MILP by removing the constraints $x_i \in \mathbb{Z}$ for $x_i \in \mathcal{D}$.

Notation

In what follows, *LP relaxation* is the linear program obtained from MILP by removing the constraints $x_i \in \mathbb{Z}$ for $x_i \in \mathcal{D}$.

Assume a global variable x^*, keeping the best solution satisfying the MILP constraints. Initialized with the undefined symbol $\perp$.

Notation

In what follows, *LP relaxation* is the linear program obtained from MILP by removing the constraints $x_i \in \mathbb{Z}$ for $x_i \in \mathcal{D}$.

Assume a global variable x^*, keeping the best solution satisfying the MILP constraints. Initialized with the undefined symbol $\perp$.

Assume a global variable f^*, keeping the value of the best solution satisfying the MILP constraints. Initialize with $f^* = \infty$.

Notation

In what follows, *LP relaxation* is the linear program obtained from MILP by removing the constraints $x_i \in \mathbb{Z}$ for $x_i \in \mathcal{D}$.

Assume a global variable x^*, keeping the best solution satisfying the MILP constraints. Initialized with the undefined symbol $\perp$.

Assume a global variable f^*, keeping the value of the best solution satisfying the MILP constraints. Initialize with $f^* = \infty$.

Keep a pool of MILP problems $\mathcal{P}$ initialized with $\mathcal{P} = \{P\}$ where P is the original MILP to be solved.

Notation

In what follows, *LP relaxation* is the linear program obtained from MILP by removing the constraints $x_i \in \mathbb{Z}$ for $x_i \in \mathcal{D}$.

Assume a global variable x^*, keeping the best solution satisfying the MILP constraints. Initialized with the undefined symbol $\perp$.

Assume a global variable f^*, keeping the value of the best solution satisfying the MILP constraints. Initialize with $f^* = \infty$.

Keep a pool of MILP problems $\mathcal{P}$ initialized with $\mathcal{P} = \{P\}$ where P is the original MILP to be solved.

In what follows, we temporarily cease to abuse notation and use $\bar{x}$ to denote the vector of values of the vector of variables x. Then $\bar{x}_i$ will denote the concrete value of the variable x_i.

Algorithm 2 Branch and Bound (Non-Deterministic)

repeat
  Choose $P \in \mathcal{P}$
  if LP relaxation of P is feasible **then**
    Find a solution $\bar{x}$ of the LP relaxation of P
    if $c^\top \bar{x} < f^*$ **then**
      if $\bar{x}_i \in \mathbb{Z}$ for all $x_i \in \mathcal{D}$ **then**
        $x^* \leftarrow \bar{x}$
        $f^* \leftarrow c^\top \bar{x}$
      else
        Choose $x_i \in \mathcal{B}$ such that $\bar{x}_i \notin \mathbb{Z}$
        Generate LP P_- by adding $x_i \leq \lfloor \bar{x}_i \rfloor$ to P
        Generate LP P_+ by adding $x_i \geq \lceil \bar{x}_i \rceil$ to P
        Add P_0 and P_1 to $\mathcal{P}$.
      end if
    end if
  end if
  $\mathcal{P} \leftarrow \mathcal{P} \setminus \{P\}$
until $\mathcal{P} = \emptyset$

Example

Consider the following MILP P:

$$
\begin{array}{ll}
\text{minimize} & -x_1 - 2x_2 - 3x_3 - 1.5x_4 \\
\text{subject to} & x_1 + x_2 + 2x_3 + 2x_4 \leq 10 \\
& 7x_1 + 8x_2 + 5x_3 + x_4 = 31.5 \\
& x_1, x_2, x_3, x_4 \geq 0
\end{array}
$$

and assume $\mathcal{D} = \{x_1, x_2, x_3\}$. That is, $x_1, x_2, x_3 \in \mathbb{Z}$.

Example

Consider the following MILP P:

$$\begin{array}{ll} \text{minimize} & -x_1 - 2x_2 - 3x_3 - 1.5x_4 \\ \text{subject to} & x_1 + x_2 + 2x_3 + 2x_4 \leq 10 \\ & 7x_1 + 8x_2 + 5x_3 + x_4 = 31.5 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{array}$$

and assume $\mathcal{D} = \{x_1, x_2, x_3\}$. That is, $x_1, x_2, x_3 \in \mathbb{Z}$.

The algorithm starts with $\mathcal{P} = \{P\}$ and $x^* = \perp$ and $f^* = \infty$.

Example

Consider the following MILP P:

$$\begin{array}{ll} \text{minimize} & -x_1 - 2x_2 - 3x_3 - 1.5x_4 \\ \text{subject to} & x_1 + x_2 + 2x_3 + 2x_4 \leq 10 \\ & 7x_1 + 8x_2 + 5x_3 + x_4 = 31.5 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{array}$$

and assume $\mathcal{D} = \{x_1, x_2, x_3\}$. That is, $x_1, x_2, x_3 \in \mathbb{Z}$.

The algorithm starts with $\mathcal{P} = \{P\}$ and $x^* = \perp$ and $f^* = \infty$.

The solution to the LP relaxation of P is:

$$x = [0, 1.1818, 4.4091, 0], \quad \text{the objective value is } -15.59$$

Example

Consider the following MILP P:

$$\begin{array}{ll} \text{minimize} & -x_1 - 2x_2 - 3x_3 - 1.5x_4 \\ \text{subject to} & x_1 + x_2 + 2x_3 + 2x_4 \leq 10 \\ & 7x_1 + 8x_2 + 5x_3 + x_4 = 31.5 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{array}$$

and assume $\mathcal{D} = \{x_1, x_2, x_3\}$. That is, $x_1, x_2, x_3 \in \mathbb{Z}$.

The algorithm starts with $\mathcal{P} = \{P\}$ and $x^* = \perp$ and $f^* = \infty$.

The solution to the LP relaxation of P is:

$$x = [0, 1.1818, 4.4091, 0], \quad \text{the objective value is } -15.59$$

Let us choose x_3. So, consider two programs:

- P_- where we add $x_3 \leq 4$ to P
- P_+ where we add $x_3 \geq 5$ to P

Now $\mathcal{P} = \{P_-, P_+\}$.

Consider first P_+.

P_+ is P with the added constraint $x_3 \geq 5$. The LP relaxation of P_+ is infeasible. We get $\mathcal{P} = \{P_-\}$.

Consider first P_+.

P_+ is P with the added constraint $x_3 \geq 5$. The LP relaxation of P_+ is infeasible. We get $\mathcal{P} = \{P_-\}$.

P_- is P with the additional constraint $x_3 \leq 4$.

Consider first P_+.

P_+ is P with the added constraint $x_3 \geq 5$. The LP relaxation of P_+ is infeasible. We get $\mathcal{P} = \{P_-\}$.

P_- is P with the additional constraint $x_3 \leq 4$.

The LP relaxation of P_- solves to

$$\bar{x} = [0, 1.4, 4, 0.3], \quad \text{the objective value is } -15.25$$

Consider first P_+.

P_+ is P with the added constraint $x_3 \geq 5$. The LP relaxation of P_+ is infeasible. We get $\mathcal{P} = \{P_-\}$.

P_- is P with the additional constraint $x_3 \leq 4$.

The LP relaxation of P_- solves to

$$\bar{x} = [0, 1.4, 4, 0.3], \quad \text{the objective value is } -15.25$$

We still have $f^* = \infty$ so we split P_- by constraining x_2:

- P_{--} is obtained from P_- by adding $x_2 \leq 1$
- P_{-+} is obtained from P_- by adding $x_2 \geq 2$

and we continue with $\mathcal{P} = \{P_{--}, P_{-+}\}$.

Consider first P_+.

P_+ is P with the added constraint $x_3 \geq 5$. The LP relaxation of P_+ is infeasible. We get $\mathcal{P} = \{P_-\}$.

P_- is P with the additional constraint $x_3 \leq 4$.

The LP relaxation of P_- solves to

$$\bar{x} = [0, 1.4, 4, 0.3], \qquad \text{the objective value is } -15.25$$

We still have $f^* = \infty$ so we split P_- by constraining x_2:

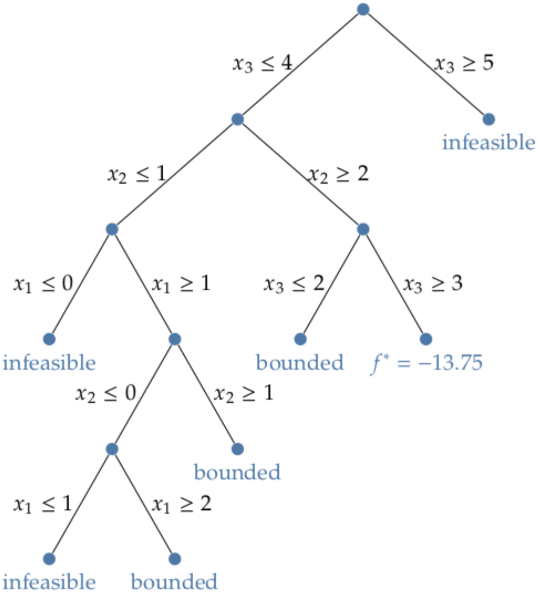
- P_{--} is obtained from P_- by adding $x_2 \leq 1$
- P_{-+} is obtained from P_- by adding $x_2 \geq 2$

and we continue with $\mathcal{P} = \{P_{--}, P_{-+}\}$.

Adding one more constraint $x_3 \geq 3$ to P_{-+} would yield a MILP solution $(0, 2, 3, 0.5)$ to the LP relaxation with the objective value equal to -13.75.

The algorithm assigns $f^* = -13.75$ and $x^* = (0, 2, 3, 0.5)$.

The remaining search always leads either to an infeasible relaxation or to a relaxation with an objective value worse than f^*.



The final solution: $x^* = (0, 2, 3, 0.5)$ and $f^* = -13.75$.

Cutting Planes

Removing Non-Integer Solutions

The basic branch and bound method generates two new problems in every step.

Removing Non-Integer Solutions

The basic branch and bound method generates two new problems in every step.

Another strategy might be to successively cut out non-integer optimal solutions and preserve the integer ones until an integer optimal solution is computed by the LP relaxation

Removing Non-Integer Solutions

The basic branch and bound method generates two new problems in every step.

Another strategy might be to successively cut out non-integer optimal solutions and preserve the integer ones until an integer optimal solution is computed by the LP relaxation

We consider a concrete method for obtaining such cuts from the ILP constraints called *Gomory cuts*.

Gomory Cuts

Consider an ILP and transform it into a MILP by adding slack variables:

$$\begin{aligned} \text{minimize} \quad & c^\top x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \\ & x \in \mathbb{Z} \text{ for } x \in \mathcal{D} \end{aligned}$$

Here, $\mathcal{D}$ contains the original (i.e., non-slack) variables of the ILP.

Gomory Cuts

Consider an ILP and transform it into a MILP by adding slack variables:

$$\begin{aligned} \text{minimize} \quad & c^\top x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \\ & x \in \mathbb{Z} \text{ for } x \in \mathcal{D} \end{aligned}$$

Here, $\mathcal{D}$ contains the original (i.e., non-slack) variables of the ILP.

We demand the integer solution only for the original $\mathcal{D}$ variables.

Gomory Cuts

Consider an ILP and transform it into a MILP by adding slack variables:

$$\begin{aligned} \text{minimize} \quad & c^\top x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \\ & x \in \mathbb{Z} \text{ for } x \in \mathcal{D} \end{aligned}$$

Here, $\mathcal{D}$ contains the original (i.e., non-slack) variables of the ILP.

We demand the integer solution only for the original $\mathcal{D}$ variables.

However, one can prove that if all constants in the ILP are integer, then there is an optimal solution where all variables (including the slacks) are integer-valued.

Gomory Cuts

Let $A = (u_1 \; \ldots, u_n)$, the basis $\{x_1, \ldots, x_n\}$, $B = (u_1 \; \ldots, u_m)$.

Gomory Cuts

Let $A = (u_1 \ \ldots, u_n)$, the basis $\{x_1, \ldots, x_n\}$, $B = (u_1 \ \ldots, u_m)$.

Consider the canonical tableau for B:

$$A' = \begin{pmatrix} 1 & \cdots & 0 & y_{1(m+1)} & \cdots & y_{1n} & b'_1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & y_{m(m+1)} & \cdots & y_{mn} & b'_m \end{pmatrix}$$

The $-z$ row is omitted as it is unnecessary for the discussion.

$$u_k = B(y_{1k}, \ldots, y_{mk})^\top \text{ for } k = 1, \ldots, n \text{ and } b' = B^{-1}b$$

Gomory Cuts

Let $A = (u_1 \ldots, u_n)$, the basis $\{x_1, \ldots, x_n\}$, $B = (u_1 \ldots, u_m)$.

Consider the canonical tableau for B:

$$A' = \begin{pmatrix} 1 & \cdots & 0 & y_{1(m+1)} & \cdots & y_{1n} & b'_1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & y_{m(m+1)} & \cdots & y_{mn} & b'_m \end{pmatrix}$$

The $-z$ row is omitted as it is unnecessary for the discussion.

$$u_k = B(y_{1k}, \ldots, y_{mk})^\top \text{ for } k = 1, \ldots, n \text{ and } b' = B^{-1}b$$

Consider a basic solution $x = (b'_1, \ldots, b'_m, 0, \ldots, 0)$.

If all $b'_1, \ldots, b'_m$ are integers, then also x solves the ILP.

Otherwise, assume that b'_i is not an integer.

Gomory Cuts

From the tableau, we know that every feasible solution x satisfies:

$$x_i + y_{i(m+1)}x_{m+1} + \cdots + y_{in}x_n = b'_i$$

Gomory Cuts

From the tableau, we know that every feasible solution x satisfies:

$$x_i + y_{i(m+1)}x_{m+1} + \cdots + y_{in}x_n = b'_i$$

Then, x also satisfies:

$$x_i + \lfloor y_{i(m+1)} \rfloor x_{m+1} + \cdots + \lfloor y_{in} \rfloor x_n \leq b'_i$$

Gomory Cuts

From the tableau, we know that every feasible solution x satisfies:

$$x_i + y_{i(m+1)} x_{m+1} + \cdots + y_{in} x_n = b'_i$$

Then, x also satisfies:

$$x_i + \lfloor y_{i(m+1)} \rfloor x_{m+1} + \cdots + \lfloor y_{in} \rfloor x_n \leq b'_i$$

Moreover, any *integer feasible solution* x satisfies:

$$x_i + \lfloor y_{i(m+1)} \rfloor x_{m+1} + \cdots + \lfloor y_{in} \rfloor x_n \leq \lfloor b'_i \rfloor$$

Gomory Cuts

From the tableau, we know that every feasible solution x satisfies:

$$x_i + y_{i(m+1)}x_{m+1} + \cdots + y_{in}x_n = b'_i$$

Then, x also satisfies:

$$x_i + \lfloor y_{i(m+1)} \rfloor x_{m+1} + \cdots + \lfloor y_{in} \rfloor x_n \leq b'_i$$

Moreover, any *integer feasible solution* x satisfies:

$$x_i + \lfloor y_{i(m+1)} \rfloor x_{m+1} + \cdots + \lfloor y_{in} \rfloor x_n \leq \lfloor b'_i \rfloor$$

But, subtracting the inequalities, integer feasible solutions x satisfy:

$$(y_{i(m+1)} - \lfloor y_{i(m+1)} \rfloor)x_{m+1} + \cdots + (y_{in} - \lfloor y_{in} \rfloor)x_n \geq b'_i - \lfloor b'_i \rfloor$$

Gomory Cuts

From the tableau, we know that every feasible solution x satisfies:

$$x_i + y_{i(m+1)}x_{m+1} + \cdots + y_{in}x_n = b'_i$$

Then, x also satisfies:

$$x_i + \lfloor y_{i(m+1)} \rfloor x_{m+1} + \cdots + \lfloor y_{in} \rfloor x_n \leq b'_i$$

Moreover, any *integer feasible solution* x satisfies:

$$x_i + \lfloor y_{i(m+1)} \rfloor x_{m+1} + \cdots + \lfloor y_{in} \rfloor x_n \leq \lfloor b'_i \rfloor$$

But, subtracting the inequalities, integer feasible solutions x satisfy:

$$(y_{i(m+1)} - \lfloor y_{i(m+1)} \rfloor)x_{m+1} + \cdots + (y_{in} - \lfloor y_{in} \rfloor)x_n \geq b'_i - \lfloor b'_i \rfloor$$

But note that the *basic feasible solution* $x = (b'_1, \ldots, b'_m, 0, \ldots, 0)$ *does not* satisfy the last inequality because $b'_i > \lfloor b'_i \rfloor$ and $x_{m+1} = \cdots = x_n = 0$.

Gomory Cuts Method

Assume that we have solved the LP and reached a basis of B.

Assume that the basic solution x w.r.t. B is non-integer.

Gomory Cuts Method

Assume that we have solved the LP and reached a basis of B.
Assume that the basic solution x w.r.t. B is non-integer.

Consider the canonical tableau for the basis B:

$$A' = \begin{pmatrix} 1 & \cdots & 0 & y_{1(m+1)} & \cdots & y_{1n} & b'_1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & y_{m(m+1)} & \cdots & y_{mn} & b'_m \end{pmatrix}$$

Gomory Cuts Method

Assume that we have solved the LP and reached a basis of B.
Assume that the basic solution x w.r.t. B is non-integer.

Consider the canonical tableau for the basis B:

$$A' = \begin{pmatrix} 1 & \cdots & 0 & y_{1(m+1)} & \cdots & y_{1n} & b'_1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & y_{m(m+1)} & \cdots & y_{mn} & b'_m \end{pmatrix}$$

Choose a non-integer component $x_i = b'_i$ of the basic feasible solution w.r.t. B

Gomory Cuts Method

Assume that we have solved the LP and reached a basis of B.
Assume that the basic solution x w.r.t. B is non-integer.

Consider the canonical tableau for the basis B:

$$A' = \begin{pmatrix} 1 & \cdots & 0 & y_{1(m+1)} & \cdots & y_{1n} & b'_1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & y_{m(m+1)} & \cdots & y_{mn} & b'_m \end{pmatrix}$$

Choose a non-integer component $x_i = b'_i$ of the basic feasible solution w.r.t. B and consider the constraint

$$x_i + (y_{i(m+1)} - \lfloor y_{i(m+1)} \rfloor) x_{m+1} + \cdots + (y_{in} - \lfloor y_{in} \rfloor) x_n \geq b'_i - \lfloor b'_i \rfloor$$

Gomory Cuts Method

Assume that we have solved the LP and reached a basis of B.
Assume that the basic solution x w.r.t. B is non-integer.

Consider the canonical tableau for the basis B:

$$A' = \begin{pmatrix} 1 & \cdots & 0 & y_{1(m+1)} & \cdots & y_{1n} & b'_1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & y_{m(m+1)} & \cdots & y_{mn} & b'_m \end{pmatrix}$$

Choose a non-integer component $x_i = b'_i$ of the basic feasible solution w.r.t. B and consider the constraint

$$x_i + (y_{i(m+1)} - \lfloor y_{i(m+1)} \rfloor) x_{m+1} + \cdots + (y_{in} - \lfloor y_{in} \rfloor) x_n \geq b'_i - \lfloor b'_i \rfloor$$

Transform the above inequality into equality by introducing a new variable x_{n+1} and obtain the following constraint (*Gomory cut*)

$$(y_{i(m+1)} - \lfloor y_{i(m+1)} \rfloor) x_{m+1} + \cdots + (y_{in} - \lfloor y_{in} \rfloor) x_n - x_{n+1} = b'_i - \lfloor b'_i \rfloor$$

Gomory Cuts Method

Assume that we have solved the LP and reached a basis of B.
Assume that the basic solution x w.r.t. B is non-integer.

Consider the canonical tableau for the basis B:

$$A' = \begin{pmatrix} 1 & \cdots & 0 & y_{1(m+1)} & \cdots & y_{1n} & b'_1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & y_{m(m+1)} & \cdots & y_{mn} & b'_m \end{pmatrix}$$

Choose a non-integer component $x_i = b'_i$ of the basic feasible solution w.r.t. B and consider the constraint

$$x_i + (y_{i(m+1)} - \lfloor y_{i(m+1)} \rfloor) x_{m+1} + \cdots + (y_{in} - \lfloor y_{in} \rfloor) x_n \geq b'_i - \lfloor b'_i \rfloor$$

Transform the above inequality into equality by introducing a new variable x_{n+1} and obtain the following constraint (*Gomory cut*)

$$(y_{i(m+1)} - \lfloor y_{i(m+1)} \rfloor) x_{m+1} + \cdots + (y_{in} - \lfloor y_{in} \rfloor) x_n - x_{n+1} = b'_i - \lfloor b'_i \rfloor$$

Add the Gomory cut and the constraint $x_{n+1} \geq 0$ to the program.

Gomory Cuts Method

Assume that we have solved the LP and reached a basis of B.
Assume that the basic solution x w.r.t. B is non-integer.

Consider the canonical tableau for the basis B:

$$A' = \begin{pmatrix} 1 & \cdots & 0 & y_{1(m+1)} & \cdots & y_{1n} & b'_1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & y_{m(m+1)} & \cdots & y_{mn} & b'_m \end{pmatrix}$$

Choose a non-integer component $x_i = b'_i$ of the basic feasible solution w.r.t. B and consider the constraint

$$x_i + (y_{i(m+1)} - \lfloor y_{i(m+1)} \rfloor) x_{m+1} + \cdots + (y_{in} - \lfloor y_{in} \rfloor) x_n \geq b'_i - \lfloor b'_i \rfloor$$

Transform the above inequality into equality by introducing a new variable x_{n+1} and obtain the following constraint (*Gomory cut*)

$$(y_{i(m+1)} - \lfloor y_{i(m+1)} \rfloor) x_{m+1} + \cdots + (y_{in} - \lfloor y_{in} \rfloor) x_n - x_{n+1} = b'_i - \lfloor b'_i \rfloor$$

Add the Gomory cut and the constraint $x_{n+1} \geq 0$ to the program.

Repeat until an integer solution is reached.

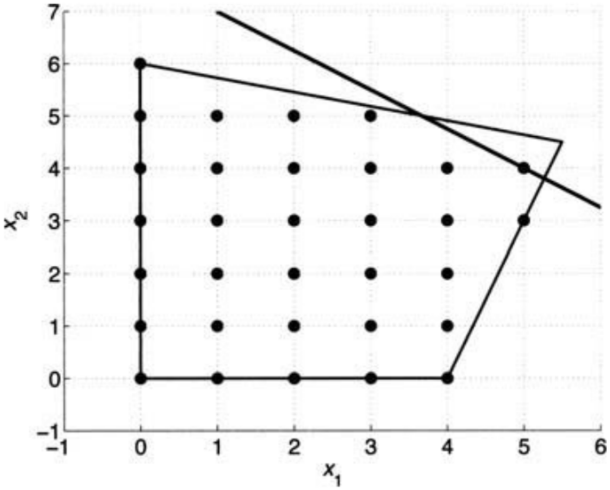
Example

Consider ILP:

$$\begin{array}{rl} \text{minimize} & -3x_1 - 4x_2 \\ \text{subject to} & 3x_1 - x_2 \leq 12 \\ & 3x_1 + 11x_2 \leq 66 \\ & x_1, x_2 \geq 0 \\ & x_1, x_2 \in \mathbb{Z} \end{array}$$

Adding slack variables x_3, x_4 we obtain the following MILP:

$$\begin{array}{rl} \text{minimize} & -3x_1 - 4x_2 \\ \text{subject to} & 3x_1 - x_2 + x_3 = 12 \\ & 3x_1 + 11x_2 + x_4 = 66 \\ & x_1, x_2, x_3, x_4 \geq 0 \\ & x_1, x_2 \in \mathbb{Z} \end{array}$$


7
6
5
4
3
2
1
0
−1
x_2
−1
0
1
2
3
4
5
6
x_1

We have

$$\begin{aligned}
\text{minimize} \quad & -3x_1 - 4x_2 \\
\text{subject to} \quad & 3x_1 - x_2 + x_3 = 12 \\
& 3x_1 + 11x_2 + x_4 = 66 \\
& x_1, x_2, x_3, x_4 \geq 0 \\
& x_1, x_2 \in \mathbb{Z}
\end{aligned}$$

We have

$$
\begin{array}{rl}
\text{minimize} & -3x_1 - 4x_2 \\
\text{subject to} & 3x_1 - x_2 + x_3 = 12 \\
& 3x_1 + 11x_2 + x_4 = 66 \\
& x_1, x_2, x_3, x_4 \geq 0 \\
& x_1, x_2 \in \mathbb{Z}
\end{array}
$$

An optimal basic solution to the LP relaxation is

$$
\left(\frac{11}{2}, \frac{9}{2}, 0, 0\right)^\top
$$

and the canonical tableau w.r.t. the basis $\{x_1, x_2\}$ is

$$
\begin{pmatrix}
x_1 & x_2 & x_3 & x_4 & b' \\
1 & 0 & \frac{11}{36} & \frac{1}{36} & \frac{11}{2} \\
0 & 1 & -\frac{1}{12} & \frac{1}{12} & \frac{9}{2}
\end{pmatrix}
$$

Let us introduce the Gomory cut corresponding to the variable x_1.

$$\begin{pmatrix} x_1 & x_2 & x_3 & x_4 & b' \\ 1 & 0 & \frac{11}{36} & \frac{1}{36} & \frac{11}{2} \\ 0 & 1 & -\frac{1}{12} & \frac{1}{12} & \frac{9}{2} \end{pmatrix}$$

Then

$$(y_{i(m+1)} - \lfloor y_{i(m+1)} \rfloor) x_{m+1} + \cdots + (y_{in} - \lfloor y_{in} \rfloor) x_n - x_{n+1} = b'_i - \lfloor b'_i \rfloor$$

with $i = 1$ and $m = 2$ turns into

$$\left(\frac{11}{36} - 0 \right) x_3 + \left(\frac{1}{36} - 0 \right) x_4 - x_5 = \frac{1}{2} \quad \left(= \frac{11}{2} - 5\right)$$

We add this constraint to our MILP.

$$
\begin{array}{rl}
\text{minimize} & -3x_1 - 4x_2 \\
\text{subject to} & 3x_1 - x_2 + x_3 = 12 \\
& 3x_1 + 11x_2 + x_4 = 66 \\
& \frac{11}{36}x_3 + \frac{1}{36}x_4 - x_5 = \frac{1}{2} \\
& x_1, x_2, x_3, x_4 \geq 0 \\
& x_1, x_2 \in \mathbb{Z}
\end{array}
$$

Solving the LP relaxation yields

$$\left(5, \frac{51}{11}, \frac{18}{11}, 0, 0\right)^\top$$

The canonical tableau for the solution is

$$
\begin{pmatrix}
x_1 & x_2 & x_3 & x_4 & x_5 & b' \\
1 & 0 & 0 & 0 & 1 & 5 \\
0 & 1 & 0 & \frac{1}{11} & -\frac{3}{11} & \frac{51}{11} \\
0 & 0 & 1 & \frac{1}{11} & -\frac{36}{11} & \frac{18}{11}
\end{pmatrix}
$$

Introduce the Gomory cut for x_2.

$$\begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & b' \\ 1 & 0 & 0 & 0 & 1 & 5 \\ 0 & 1 & 0 & \frac{1}{11} & -\frac{3}{11} & \frac{51}{11} \\ 0 & 0 & 1 & \frac{1}{11} & -\frac{36}{11} & \frac{18}{11} \end{pmatrix}$$

Then

$$(y_{i(m+1)} - \lfloor y_{i(m+1)} \rfloor) x_{m+1} + \cdots + (y_{in} - \lfloor y_{in} \rfloor) x_n - x_{n+1} = b'_i - \lfloor b'_i \rfloor$$

with $i = 2$ and $m = 3$ turns into

$$\left(\frac{1}{11} - 0 \right) x_4 + \left(-\frac{3}{11} + \frac{11}{11} \right) x_5 - x_6 = \frac{7}{11} \quad (= \frac{51}{11} - \frac{44}{11})$$

We add this to our MILP.

$$
\begin{aligned}
\text{minimize} \quad & -3x_1 - 4x_2 \\
\text{subject to} \quad & 3x_1 - x_2 + x_3 = 12 \\
& 3x_1 + 11x_2 + x_4 = 66 \\
& \tfrac{11}{36}x_3 + \tfrac{1}{36}x_4 - x_5 = \tfrac{1}{2} \\
& \tfrac{1}{11}x_4 + \tfrac{8}{11}x_5 - x_6 = \tfrac{7}{11} \\
& x_1, x_2, x_3, x_4 \geq 0 \\
& x_1, x_2 \in \mathbb{Z}
\end{aligned}
$$

Once more the solution of the above is non-integer. However, introducing another Gomory cut (and a variable x_7) would yield a solution:

$$(5, 4, 1, 7, 0, 0, 0)^\top$$

Which gives the point $(x_1, x_2) = (5, 4)$ corresponding to the graphical solution.

Cutting Planes Technique

The method based on Gomory cuts was one of the first solutions to the integer linear programming problem with proven convergence (in the 1950s).

Cutting Planes Technique

The method based on Gomory cuts was one of the first solutions to the integer linear programming problem with proven convergence (in the 1950s).

The convergence rate is unsatisfactory in practice; many more methods have been devised based on algebraic principles (combinations of inequalities and rounding), geometry, etc.

Cutting Planes Technique

The method based on Gomory cuts was one of the first solutions to the integer linear programming problem with proven convergence (in the 1950s).

The convergence rate is unsatisfactory in practice; many more methods have been devised based on algebraic principles (combinations of inequalities and rounding), geometry, etc.

Cutting planes are also used in other non-linear, non-smooth optimization methods.

Cutting Planes Technique

The method based on Gomory cuts was one of the first solutions to the integer linear programming problem with proven convergence (in the 1950s).

The convergence rate is unsatisfactory in practice; many more methods have been devised based on algebraic principles (combinations of inequalities and rounding), geometry, etc.

Cutting planes are also used in other non-linear, non-smooth optimization methods.

Most importantly, cutting plane techniques are combined with branch and bound methods. The constraints are introduced before branching to eliminate some solutions before the split.

The resulting method is called *branch and cut*.

Summary of Integer Linear Programming

We have considered:

- Linear Programming (LP)
  Linear objective and constraints.

Summary of Integer Linear Programming

We have considered:

- Linear Programming (LP)
  Linear objective and constraints.
- 0-1 Integer Linear Programming (0-1 ILP)
  Linear objective and constraints. All variables restricted to $\{0, 1\}$.

Summary of Integer Linear Programming

We have considered:

- Linear Programming (LP)
  Linear objective and constraints.
- 0-1 Integer Linear Programming (0-1 ILP)
  Linear objective and constraints. All variables restricted to $\{0, 1\}$.
- 0-1 Mixed Integer Programming (0-1 MILP)
  Linear objective and constraints. Some variables restricted to $\{0, 1\}$.

Summary of Integer Linear Programming

We have considered:

- Linear Programming (LP)
  Linear objective and constraints.
- 0-1 Integer Linear Programming (0-1 ILP)
  Linear objective and constraints. All variables restricted to $\{0, 1\}$.
- 0-1 Mixed Integer Programming (0-1 MILP)
  Linear objective and constraints. Some variables restricted to $\{0, 1\}$.
- Integer Linear Programming (ILP)
  Linear objective and constraints. All variables restricted to $\mathbb{Z}$.

Summary of Integer Linear Programming

We have considered:

- Linear Programming (LP)
  Linear objective and constraints.
- 0-1 Integer Linear Programming (0-1 ILP)
  Linear objective and constraints. All variables restricted to $\{0, 1\}$.
- 0-1 Mixed Integer Programming (0-1 MILP)
  Linear objective and constraints. Some variables restricted to $\{0, 1\}$.
- Integer Linear Programming (ILP)
  Linear objective and constraints. All variables restricted to $\mathbb{Z}$.
- Mixed Integer Linear Programming (MILP)
  Linear objective and constraints. Some variables restricted to $\mathbb{Z}$.

Summary of Integer Linear Programming

Complexity:

- Even the 0-1 Integer Linear Programming is NP-hard.

  Linear programming is in $\mathbb{P}$-time.

Summary of Integer Linear Programming

Complexity:

- Even the 0-1 Integer Linear Programming is NP-hard.
  Linear programming is in $\mathbb{P}$-time.

Algorithms:

- Branch and Bound
  - 0-1 MILP: Search through possible assignments of 0 and 1 to some discrete variables while solving the LP relaxations
    Branching with the choice of 0/1 values of variables, bounding with a solution found so far.

Summary of Integer Linear Programming

Complexity:

- Even the 0-1 Integer Linear Programming is NP-hard.
  Linear programming is in $\mathbb{P}$-time.

Algorithms:

- Branch and Bound
  - 0-1 MILP: Search through possible assignments of 0 and 1 to some discrete variables while solving the LP relaxations
    Branching with the choice of 0/1 values of variables, bounding with a solution found so far.
  - MILP: Solve LP relaxation, use non-integer values of the solution to introduce constraints, removing such values from the solution.

Summary of Integer Linear Programming

Complexity:

- Even the 0-1 Integer Linear Programming is NP-hard.
  Linear programming is in $\mathbb{P}$-time.

Algorithms:

- Branch and Bound
  - 0-1 MILP: Search through possible assignments of 0 and 1 to some discrete variables while solving the LP relaxations
    Branching with the choice of 0/1 values of variables, bounding with a solution found so far.
  - MILP: Solve LP relaxation, use non-integer values of the solution to introduce constraints, removing such values from the solution.
- Cutting planes
  - Sequentially cut out portions of the LP relaxation feasible space by introducing cuts based on solutions of LP relaxations.

Summary of Integer Linear Programming

Complexity:

- Even the 0-1 Integer Linear Programming is NP-hard.

  Linear programming is in $\mathbb{P}$-time.

Algorithms:

- Branch and Bound
  - 0-1 MILP: Search through possible assignments of 0 and 1 to some discrete variables while solving the LP relaxations

    Branching with the choice of 0/1 values of variables, bounding with a solution found so far.
  - MILP: Solve LP relaxation, use non-integer values of the solution to introduce constraints, removing such values from the solution.
- Cutting planes
  - Sequentially cut out portions of the LP relaxation feasible space by introducing cuts based on solutions of LP relaxations.
  - Does not branch but is usually combined with branch and bound (branch and cut).