Lecture 9

Linear programming (II) – simplex method

Emil Gustavsson
Fraunhofer-Chalmers Centre
November 23, 2017

CHALMERS | GÖTEBORGS UNIVERSITET

Linear programs (LP) in standard form

- Consider LP in standard form

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & \boldsymbol{c}^T \boldsymbol{x}, \\ \text{subject to} \quad & Ax = b, \\ & x \geq \mathbf{0}. \end{aligned}$$

- $A \in \mathbb{R}^{m \times n}$ is a given matrix, and b is a given vector,
- $\text{rank}(A) = m$, $b \geq \mathbf{0}$.

Basic feasible solution (BFS)

Standard form polyhedron $P = \{x \mid Ax = b, x \geq \mathbf{0}\}$, $A \in \mathbb{R}^{m \times n}$, $\operatorname{rank}(A) = m$

A point $\bar{x}$ is a **basic feasible solution** (BFS) if it is a basic solution that is feasible. That is, $\bar{x}$ is a BFS if

- $\bar{x} \geq \mathbf{0}$,
- $\boldsymbol{A\bar{x}} = \boldsymbol{b}$, and
- the columns of $\boldsymbol{A}$ corresponding to non-zero components of $\boldsymbol{\bar{x}}$ are linearly independent

Basic and non-basic variables

For any BFS $\bar{\boldsymbol{x}}$, we can reorder the variables according to

$$\bar{\boldsymbol{x}} = \begin{pmatrix} \boldsymbol{x}_B \\ \boldsymbol{x}_N \end{pmatrix}, \quad \boldsymbol{A} = (\boldsymbol{B}, \boldsymbol{N}), \quad \boldsymbol{c} = \begin{pmatrix} \boldsymbol{c}_B \\ \boldsymbol{c}_N \end{pmatrix},$$

such that

- $B \in \mathbb{R}^{m \times m}$, $\text{rank}(B) = m$.
- $x_N = \mathbf{0}^{n-m}$.
- $x_B = B^{-1}b$ (as a consequence of $A\bar{x} = Bx_B + Nx_N = b$).

We call

- x_B the **basic variables**. If $x_B \not> \mathbf{0}$ then BFS $\bar{x}$ is called **degenerate**.
- x_N the **non-basic variables**.
- B the **basis matrix**. Each BFS is associated with at least one basis.

Summary from last time

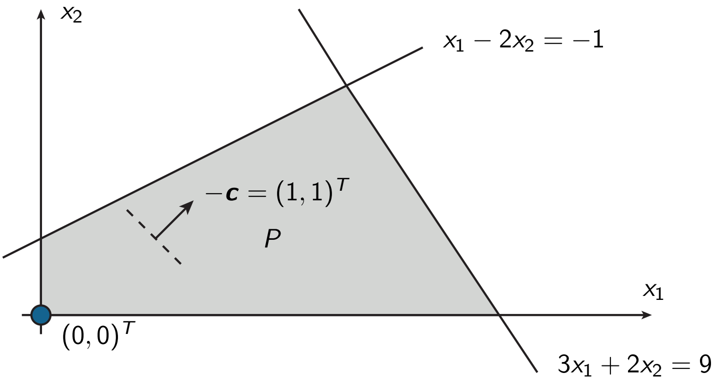
So far, we have seen

- All linear programs can be written in standard form.
- Extreme point = basic feasible solution (BFS).
- If a standard form LP has finite optimal solution, then it has an optimal BFS.

We solve standard form LP by searching only the BFS's. This is the main characteristic of the **simplex algorithm**.

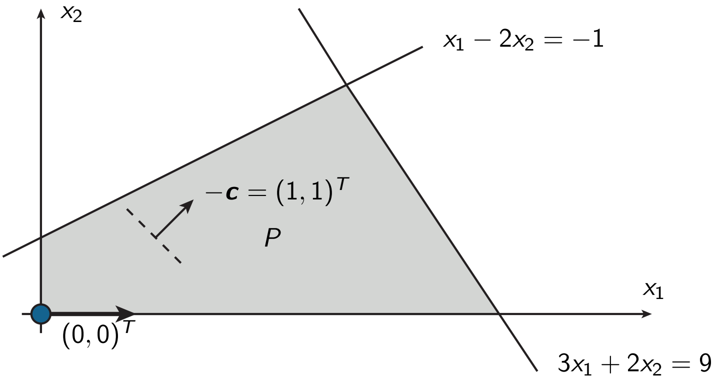
Graphic illustration

Start at a BFS, in this case $(0,0)^T$.



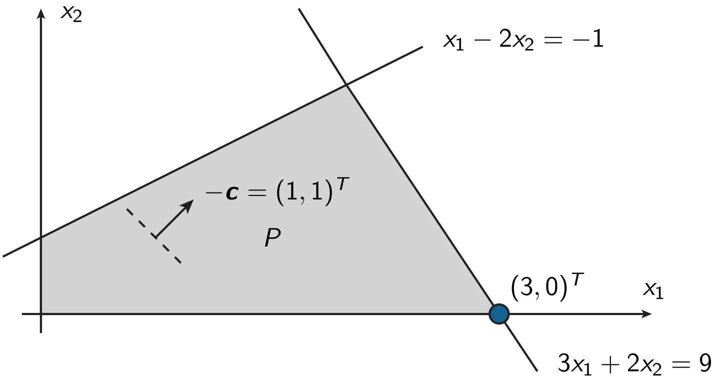
Graphic illustration

Find a feasible descent direction towards an adjacent BFS.



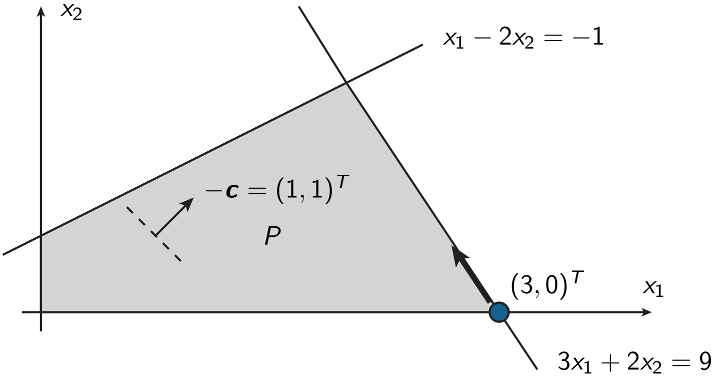
Graphic illustration

Move along the search direction until a new BFS is found.

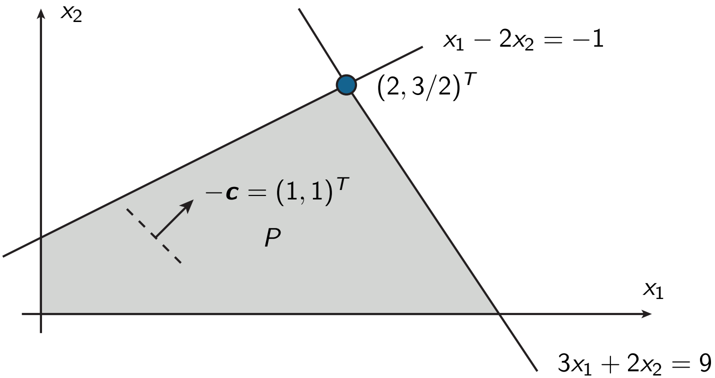


Graphic illustration

Find a new feasible descent direction at the current BFS.

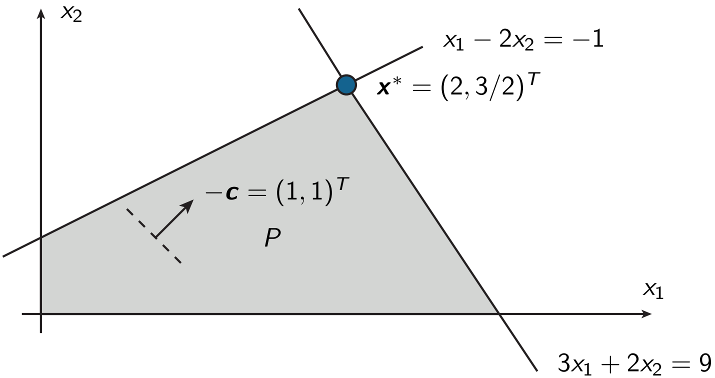


Move along the search direction.



Graphic illustration

If no feasible descent directions exist, the current BFS is declared optimal.



Developing the simplex algorithm

To develop the simplex algorithm, we translate geometric picture into algebraic calculations. We need to...

1. Determine whether or not current BFS is optimal.
2. Find a feasible descent direction at any BFS.
3. Determine the step-size to move along a feasible descent direction.
4. Update iterate, and go back to step 1.

We will discuss in this order: 2, 3, 4, 1.

Search direction

- Simplex method updates iterate according to: $\bar{x} \leftarrow \bar{x} + \theta d$
  - d is search direction, to be discussed
  - $\theta \geq 0$ is step-size, to be discussed
- At BFS $\bar{x} = \begin{pmatrix} x_B \\ x_N \end{pmatrix}$ with $A = (B, N)$; partition search dir $d = \begin{pmatrix} d_B \\ d_N \end{pmatrix}$.
- In simplex method, we **update one non-basic variable at a time**

$$d_N = e_j, \quad e_j \text{ is the } j\text{-th unit vector in } \mathbb{R}^{n-m} \text{ for } j = 1, \ldots, n-m$$

- d_B is not arbitrary – it is decided by feasibility of $\bar{x} + \theta d$:
  $A(\bar{x} + \theta d) = b \implies Ad = 0 \xRightarrow{A=(B,N)} d_B = -B^{-1}Ne_j = -B^{-1}N_j$

We consider search directions: $d_j = \begin{pmatrix} -B^{-1}N_j \\ e_j \end{pmatrix}, \; j = 1, \ldots, n-m$

Search direction, numerical example

$$B = \begin{pmatrix} 1 & 1 \\ 2 & 0 \end{pmatrix}, \; N = \begin{pmatrix} 1 & 1 \\ 3 & 4 \end{pmatrix}, \; B^{-1} = \begin{pmatrix} 0 & 1/2 \\ 1 & -1/2 \end{pmatrix}, \; N_1 = \begin{pmatrix} 1 \\ 3 \end{pmatrix}, \; N_2 = \begin{pmatrix} 1 \\ 4 \end{pmatrix}$$

First search direction:

$$d_1 = \begin{pmatrix} -B^{-1}N_1 \\ e_1 \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} 0 & -1/2 \\ -1 & 1/2 \end{pmatrix} \begin{pmatrix} 1 \\ 3 \end{pmatrix} \\ \begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} -3/2 \\ 1/2 \\ 1 \\ 0 \end{pmatrix}$$

Second search direction:

$$d_2 = \begin{pmatrix} -B^{-1}N_2 \\ e_2 \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} 0 & -1/2 \\ -1 & 1/2 \end{pmatrix} \begin{pmatrix} 1 \\ 4 \end{pmatrix} \\ \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} -2 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

Reduced costs

- From $\bar{x}$ to $\bar{x} + \theta d_j$, objective value change is

$$c^T(\bar{x} + \theta d_j - \bar{x}) = \theta \cdot c^T d_j = \theta \cdot (c_B^T, c_N^T) \begin{pmatrix} -B^{-1}N_j \\ e_j \end{pmatrix} := \theta \cdot (\tilde{c}_N)_j$$

$(\tilde{c}_N)_j := (c_N^T - c_B^T B^{-1} N)_j$ is the **reduced cost** for non-basic var $(x_N)_j$

$\tilde{c}_N := (c_N^T - c_B^T B^{-1} N)^T$ are **reduced costs** for all non-basic variables

- If $(\tilde{c}_N)_j \geq 0$, d_j does not decrease objective value.
- If $(\tilde{c}_N)_j < 0$, consider update $\bar{x} + \theta d_j$ with θ as large as possible since objective value change is $\theta \cdot (\tilde{c}_N)_j < 0$ as long as $\theta > 0$.

Reduced costs, numerical example

$$B = \begin{pmatrix} 1 & 1 \\ 2 & 0 \end{pmatrix},\ N = \begin{pmatrix} 1 & 1 \\ 3 & 4 \end{pmatrix},\ B^{-1} = \begin{pmatrix} 0 & 1/2 \\ 1 & -1/2 \end{pmatrix},\ c^T = (\underbrace{1, -1}_{c_B^T}, \underbrace{3, 0}_{c_N^T})$$

Reduced costs (for non-basic variables) are

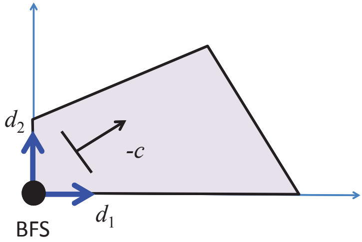
$$\begin{aligned} \tilde{c}_N^T = (c_N^T - c_B^T B^{-1} N) &= (3, 0) - (1, -1) \begin{pmatrix} 0 & 1/2 \\ 1 & -1/2 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 3 & 4 \end{pmatrix} \\ &= (1, -3) \end{aligned}$$

- Search direction d_1 does not decrease objective value;
- Search direction d_2 may decrease objective value.

Reduced costs, in picture

Reduced cost $(\tilde{c}_N)_j = (c_N^T - c_B^T B^{-1} N)_j = c^T d_j$

- $(\tilde{c}_N)_j$ is inner product of cost vector c and direction d_j.
- $(\tilde{c}_N)_j < 0 \implies d_j$ has positive projection along $-c$.



Question: Iterate moves along d_j with $(\tilde{c}_N)_j < 0$, but how far?

Update along search direction

At BFS $\bar{x} = (x_B^T, \mathbf{0})^T$, negative reduced cost for $(x_N)_j$ (i.e., $(\tilde{c}_N)_j < 0$).

- Iterate update
  $$\bar{x} + \theta d_j = \begin{pmatrix} x_B \\ \mathbf{0} \end{pmatrix} + \theta \begin{pmatrix} d_B \\ d_N \end{pmatrix} = \begin{pmatrix} x_B - \theta B^{-1} N_j \\ \theta e_j \end{pmatrix}, \ \theta \geq 0$$
- If $B^{-1} N_j \leq 0$ then $\bar{x} + \theta d_j \geq 0$ for all $\theta \geq 0$. Let $\theta \to \infty$, and we conclude that objective value is **unbounded from below**.
- If $B^{-1} N_j \not\leq 0$ some entry of $x_B - \theta B^{-1} N_j$ becomes 0 as θ increases.

$$\theta \leq \theta^* = \min_{k:(B^{-1}N_j)_k > 0} \frac{(x_B)_k}{(B^{-1}N_j)_k}, \text{ and let } i \text{ be s.t. } \theta^* = \frac{(x_B)_i}{(B^{-1}N_j)_i}.$$

- Thus, we arrive at new iterate $\bar{x} + \theta^* d_j$ with $(x_B - \theta^* B^{-1} N_j)_i = 0$. Note: θ^* can be zero if $\bar{x}$ is degenerate!

Example, continued (1)

BFS $\bar{x} = (1,1,0,0)$, $B = \begin{pmatrix} 1 & 1 \\ 2 & 0 \end{pmatrix}$, $N = \begin{pmatrix} 1 & 1 \\ 3 & 4 \end{pmatrix}$, $c_B = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$, $c_N = \begin{pmatrix} 3 \\ 0 \end{pmatrix}$

Reduced costs:

$$\tilde{c}_N^T = (c_N^T - c_B^T B^{-1} N) = (3,0) - (1,-1) \begin{pmatrix} 0 & 1/2 \\ 1 & -1/2 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 3 & 4 \end{pmatrix} = (1,-3)$$

Search direction $d_2 = (-2,1,0,1)^T$ may reduce objective value.

For d_2, updated iterate $\bar{x} + \theta d_2$ is

$$\begin{pmatrix} x_B - \theta B^{-1} N_2 \\ \theta e_2 \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} - \theta \begin{pmatrix} 0 & 1/2 \\ 1 & -1/2 \end{pmatrix} \begin{pmatrix} 1 \\ 4 \end{pmatrix} \\ \theta \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} - \theta \begin{pmatrix} 2 \\ -1 \end{pmatrix} \\ \theta \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{pmatrix}$$

$\implies$ max step-size $\theta^* = \frac{(x_B)_1}{(B^{-1}N_2)_1} = \frac{1}{2}$

Example, continued (2)

- Original BFS

$$\bar{x} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

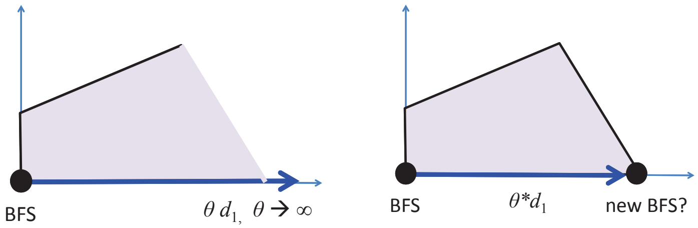
- Updated iterate with θ^*

$$\bar{x} + \theta^* d_2 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} + (1/2) \begin{pmatrix} -2 \\ 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ 3/2 \\ 0 \\ \mathbf{1/2} \end{pmatrix}$$

- First basic variable turns 0, second non-basic variable turns positive

Iterate update, in picture

Updating $\bar{x} + \theta d_j$ either tells us objective value is unbounded (left picture), or a possibly new point $\bar{x} + \theta^* d_j$ is reached (right picture).



Question: What is $\bar{x} + \theta^* d_j$? Is it a BFS? How is it related to $\bar{x}$?

Change of basis

From $\bar{x}$ to $\bar{x} + \theta^* d_j$, the i-th basic variable $(x_B)_i$ becomes zero, whereas j-th non-basic variable $(x_N)_j$ (i.e., the $(m+j)$-th variable) becomes θ^*:

$$\bar{x} = \begin{pmatrix} \vdots \\ (x_B)_i \\ \vdots \\ \mathbf{0} \end{pmatrix} \qquad \bar{x} + \theta^* d_j = \begin{pmatrix} \vdots \\ 0 \\ \vdots \\ \theta^* e_j \end{pmatrix}$$

- Can show the columns $a_1, \ldots, a_{i-1}, a_{m+j}, a_{i+1}, \ldots, a_m$ are linearly independent (forming a new basis), and $\bar{x} + \theta^* d_j$ is indeed a BFS.
- We say $(x_B)_i$ **leaves the basis** to become non-basic variable, whereas $(x_N)_j$ **enters the basis** to become basic variable.
- Prop 8.13 in text shows $\bar{x}$ and $\bar{x} + \theta^* d_j$ are in fact **adjacent** BFS's.

To recap so far

We have seen so far...

- At a BFS with $A = (B, N)$, compute search directions $d_j = \begin{pmatrix} -B^{-1}N_j \\ e_j \end{pmatrix}$, $j = 1, \dots, n-m$.
- Evaluate the reduced costs $\tilde{c}_N := (c_N^T - c_B^T B^{-1} N)^T$ to see which directions are profitable (which non-basic variable to enter basis).
- The consequence of updating $\bar{x} + \theta d_j$ for some d_j with $(\tilde{c}_N)_j < 0$ – either objective value is unbounded or an adjacent BFS is reached.

But...

- What if $\tilde{c}_N \geq \mathbf{0}$, as all our (considered) search directions are not profitable?

Nonnegative reduced costs imply optimality:

> **Theorem**
> Let $\bar{x}$ be a BFS associated with basis matrix B, and let $\tilde{c}_N = (c_N^T - c_B^T B^{-1} N)^T$ be the corresponding vector of reduced costs for the non-basic variables. If $\tilde{c}_N \geq 0$, then $\bar{x}$ is optimal.

Proof: All feasible directions d at $\bar{x} = (x_B^T, x_N^T)^T$ are of the form

$$d = \begin{pmatrix} -B^{-1}N\, d_N \\ d_N \end{pmatrix} \implies c^T d = \underbrace{(c_N^T - c_B^T B^{-1} N)}_{\tilde{c}_N^T} d_N$$

$\tilde{c}_N \geq 0$ and $d_N \geq 0$ (i.e., feasible direction) implies $c^T d \geq 0$.

The conceptual simplex algorithm

1. Assume we have an initial BFS $\bar{x} = (x_B^T, x_N^T)^T$ with $A = (B, N)$.
2. Compute reduced costs $\tilde{c}_N = (c_N^T - c_B^T B^{-1} N)^T$.
   - If $\tilde{c}_N \geq \mathbf{0}$, then current BFS is optimal, terminate.
   - If $\tilde{c}_N \not\geq \mathbf{0}$, choose some non-basic variable index j^* s.t. $(\tilde{c}_N)_j^* < 0$ as **incoming variable**.
3. Compute $B^{-1} N_j$.
   - If $B^{-1} N_j \leq 0$, then objective value is $-\infty$, terminate.
   - If $B^{-1} N_j \not\leq 0$, compute $\theta^* = \min_{k:(B^{-1}N_j)_k > 0} \frac{(x_B)_k}{(B^{-1}N_j)_k}$ and let i^* be an index solving that problem. Let i^* be the **outgoing variable**
4. Update basis. Replace variable j^* in the basis with variable i^*. Go to step 1.

Simplex algorithm terminates in finite number of steps if all BFS's are non-degenerate.

Theorem

If feasible set is nonempty and every BFS is non-degenerate, then the simplex algorithm terminates in finite number of iterations. At termination, two possibilities are allowed:
(a) an optimal basis B found with the associated optimal BFS.
(b) a direction d found s.t. $Ad = 0$, $d \geq 0$ and $c^T d < 0$, thus optimal objective value is $-\infty$.

Simplex algorithm + cycle-breaking rule (e.g. Bland's rule) $\Longrightarrow$ finite termination even with degenerate BFS.

Computational complexity

- The simplex algorithm works very well in practice.
- The simplex algorithm can, in the worst case, visits all $\binom{n}{m}$ BFS's before termination – worst-case computation effort is exponential.
- Polynomial-time algorithms are available (e.g., ellipsoid algorithm, interior point algorithms). See coming lectures.

The initial BFS assumption

- So far, we assume we know an initial BFS to start simplex method.

  Q: How do we find an initial BFS?

  A: We are lucky if the LP is of the special form

$$
\begin{aligned}
\underset{\boldsymbol{x},\boldsymbol{y}}{\text{minimize}} \quad & c_x^T x + c_y^T y, \\
\text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} + \boldsymbol{I}^m y = \boldsymbol{b}, \\
& \boldsymbol{x} \geq \mathbf{0}^n, \\
& \boldsymbol{y} \geq \mathbf{0}^m.
\end{aligned}
$$

- We can let $(\boldsymbol{x} = \mathbf{0}^n, \boldsymbol{y} = \boldsymbol{b})^T$ be the initial BFS with initial basis $\boldsymbol{I}^m$.

  Q: What if our LP is not of the special form?

  A: We create one by considering the **Phase-I problem**.

Phase-I problem for initial BFS

Phase-I problem introduces **artificial variables** a_i in every row.

$$\begin{aligned} w^* = \text{minimize } \; & w = (\mathbf{1}^m)^T \boldsymbol{a}, \\ \text{subject to } \; & \boldsymbol{A}\boldsymbol{x} + \boldsymbol{I}^m \boldsymbol{a} = \boldsymbol{b}, \\ & \boldsymbol{x} \geq \mathbf{0}^n, \\ & \boldsymbol{a} \geq \mathbf{0}^m. \end{aligned}$$

- Why is this easier? Because $\boldsymbol{a} = \boldsymbol{b}$, $\boldsymbol{x} = \mathbf{0}^n$ is an initial BFS.
- $w^* = 0 \implies$ Optimal solution $\boldsymbol{a}^* = \mathbf{0}^m$
  $\boldsymbol{x}^*$ BFS in the original problem
- $w^* > 0 \implies$ There is no BFS to the original problem
  The original problem is infeasible