The Revised Simplex Method

Combinatorial Problem Solving (CPS)

Javier Larrosa    Albert Oliveras    Enric Rodríguez-Carbonell

March 24, 2023
The simplex method we have seen so far is called **tableau simplex method**

Some observations:

- At each iteration we **update** the full tableau
  
  \[ x_B = B^{-1}b - B^{-1}Rx_R \]

  for the new basis

- But ...
  
  - For **pricing** only one negative reduced cost is needed
  
  - For **ratio test**, only
    
    - the **column** of the chosen non-basic variable in the tableau, and
    
    - the current basic **solution** are needed
Revised Simplex Method

- **Idea:** do not keep a representation of the full tableau, only $B^{-1}$
- **Advantages** over the tableau version:
  - **Time** and **space** are saved
  - **Errors** due to floating-point arithmetic are easier to **control**
Revised Simplex Method

- **Idea:** do not keep a representation of the full tableau, only $B^{-1}$
- **Advantages** over the tableau version:
  - Time and space are saved
  - Errors due to floating-point arithmetic are easier to control

- We will revise the algorithm and express it in terms of $B^{-1}$
- For simplicity, let us revise the version of the algorithm for LPs of the form

$$\begin{align*}
\min \ z &= c^T x \\
Ax &= b \\
x &\geq 0
\end{align*}$$
Let us see how the basic solution is expressed in terms of $B^{-1}$.

For any basis $B$, values of basic variables can be expressed in terms of non-basic variables:

$$Bx_B + Rx_R = b$$
$$Bx_B = b - Rx_R$$
$$x_B = B^{-1}b - B^{-1}Rx_R$$
Basic Solution

- Let us see how the basic solution is expressed in terms of $B^{-1}$.

- For any basis $B$, values of basic variables can be expressed in terms of non-basic variables:

\[
Bx_B + Rx_R = b \\
Bx_B = b - Rx_R \\
x_B = B^{-1}b - B^{-1}Rx_R
\]

- By definition, the basic solution corresponds to assigning null values to all non-basic variables: $x_R = 0$
  Then $x_B = B^{-1}b$.

- We will denote the basic solution (projected on basic variables) with $\beta := B^{-1}b$. 

Optimality Condition

- Let us see now how to express the reduced costs in terms of $B^{-1}$

- Recall the equation of basic variables in terms of non-basic variables:

$$x_B = B^{-1}b - B^{-1}Rx_R$$

- Cost function can be split: $c^T x = c_B^T x_B + c_R^T x_R$, where $c_B^T$ are the costs of basic variables,
  $c_R^T$ are the costs of non-basic variables
Optimality Condition

- Let us see now how to express the reduced costs in terms of $B^{-1}$
- Recall the equation of basic variables in terms of non-basic variables:
  $$x_B = B^{-1}b - B^{-1}Rx_R$$
- Cost function can be split: $c^T x = c^T_B x_B + c^T_R x_R$, where
  - $c^T_B$ are the costs of basic variables,
  - $c^T_R$ are the costs of non-basic variables
- We can express the cost function in terms of non-basic variables:
  $$c^T x = c^T_B x_B + c^T_R x_R =
  c^T_B (B^{-1}b - B^{-1}Rx_R) + c^T_R x_R =
  c^T_B B^{-1}b - c^T_B B^{-1}Rx_R + c^T_R x_R =
  c^T_B B^{-1}b + (c^T_R - c^T_B B^{-1}R)x_R$$
We found that $c^T x = c_B^T B^{-1} b + (c_R^T - c_B^T B^{-1} R)x_R$

The part that depends on non-basic variables is $(c_R^T - c_B^T B^{-1} R)x_R$

Let $a_j$ be the column in $A$ corresponding to variable $x_j \in x_R$.

The coefficient of $x_j$ in $(c_R^T - c_B^T B^{-1} R)x_R$ is $c_j - c_B^T B^{-1} a_j$

We will denote the reduced cost of $x_j$ with $d_j := c_j - c_B^T B^{-1} a_j$

Optimality condition: $d_j \geq 0$ for all $j \in \mathcal{R}$
Cost at Basic Solution

Let's see how to express the value of the cost function at the basic solution.

We found that $c^T x = c_B^T B^{-1} b + d^T_R x_R$, where $d_j = c_j - c_B^T B^{-1} a_j$.

We will denote the value of the cost function at the basic solution with $z$.

Taking $x_R = 0$ in the above equation: $z := c_B^T B^{-1} b$. 
Let's see how to express the value of the cost function at the basic solution

We found that \( c^T x = c_B^T B^{-1} b + d_R^T x_R \), where \( d_j = c_j - c_B^T B^{-1} a_j \)

We will denote the value of the cost function at the basic solution with \( z \)

Taking \( x_R = 0 \) in the above equation: \( z := c_B^T B^{-1} b \)

To avoid repeating computations:

Let us define the simplex multiplier as \( \pi := (B^T)^{-1} c_B \)

Then \( \pi^T = c_B^T B^{-1} \)

So \( d_j = c_j - \pi^T a_j \)

(And \( z = \pi^T b \))
Improving a Non-Optimal Solution

- Let us assume the optimality condition is violated.

- Let \( x_q \) be a non-basic variable such that its reduced cost is \( d_q < 0 \).

- Current value of \( x_q \) is 0. We can improve by increasing only this value while non-negativity constraints of basic variables are satisfied.
Improving a Non-Optimal Solution

- Let us assume the optimality condition is violated

- Let $x_q$ be a non-basic variable such that its reduced cost is $d_q < 0$

- Current value of $x_q$ is 0.
  We can **improve** by **increasing** only this value while non-negativity constraints of basic variables are satisfied.

- Let $t \geq 0$ be the new value for $x_q$.
  Let $x_B(t)$ be the values of basic variables in terms of $t$
  Let $x_R(t)$ be the values of non-basic variables in terms of $t$
  Note that $x_q(t) = t$, and $x_p(t) = 0$ if $p \in R$ and $p \neq q$
Let us assume the optimality condition is violated

Let $x_q$ be a non-basic variable such that its reduced cost is $d_q < 0$

Current value of $x_q$ is $0$.
We can improve by increasing only this value while non-negativity constraints of basic variables are satisfied.

Let $t \geq 0$ be the new value for $x_q$.
Let $x_B(t)$ be the values of basic variables in terms of $t$
Let $x_R(t)$ be the values of non-basic variables in terms of $t$
Note that $x_q(t) = t$, and $x_p(t) = 0$ if $p \in R$ and $p \neq q$

So $x_B(t) = B^{-1}b - B^{-1}Rx_R(t) = B^{-1}b - B^{-1}a_q t = \beta - t\alpha_q$

where $\beta = B^{-1}b$ is the basic solution
and we denote the column in the tableau of $x_q$ as $\alpha_q := B^{-1}a_q$
Improving a Non-Optimal Solution

■ How much do we improve?

How does the objective value change as a function of $t$?

$$z(t) =$$

$$c^T x(t) =$$

$$c_B^T x_B(t) + c_R^T x_R(t) =$$

$$c_B^T x_B(t) + c_q t =$$

$$c_B^T \beta - t c_B^T \alpha_q + c_q t =$$

$$c_B^T \beta - t c_B^T B^{-1} a_q + c_q t =$$

$$z + td_q$$

■ As expected, the improvement in cost is $\Delta z = z(t) - z = td_q$
Improving a Non-Optimal Solution

- Recall that $x_B(t) = \beta - t\alpha_q$
- How can we satisfy the non-negativity constraints of basic variables?
Improving a Non-Optimal Solution

- Recall that \( x_B(t) = \beta - t\alpha_q \)
- How can we satisfy the non-negativity constraints of basic variables?
- Basic variables have indices \( B = (k_1, ..., k_m) \)
- Let \( i \in \{1, ..., m\} \). The \( i \)-th basic variable is \( x_{k_i} \)
- Value of \( x_{k_i} \) as a function of \( t \) is the \( i \)-th component of \( x_B(t): \beta_i - t\alpha^i_q \), where \( \beta_i \) is the \( i \)-th component of \( \beta \) and \( \alpha^i_q \) is the \( i \)-th component of \( \alpha_q \)
Improving a Non-Optimal Solution

- Recall that $x_B(t) = \beta - t\alpha_q$
- How can we satisfy the non-negativity constraints of basic variables?
- Basic variables have indices $B = (k_1, \ldots, k_m)$
- Let $i \in \{1, \ldots, m\}$. The $i$-th basic variable is $x_{k_i}$
- Value of $x_{k_i}$ as a function of $t$ is the $i$-th component of $x_B(t) = \beta_i - t\alpha^i_q$, where $\beta_i$ is the $i$-th component of $\beta$ and $\alpha^i_q$ is the $i$-th component of $\alpha_q$
- We need $\beta_i - t\alpha^i_q \geq 0 \iff \beta_i \geq t\alpha^i_q$
  - If $\alpha^i_q \leq 0$ the constraint is satisfied for all $t \geq 0$
  - If $\alpha^i_q > 0$ we need $\frac{\beta_i}{\alpha^i_q} \geq t$
Improving a Non-Optimal Solution

- Recall that $x_B(t) = \beta - t\alpha_q$
- How can we satisfy the non-negativity constraints of basic variables?
- Basic variables have indices $B = (k_1, ..., k_m)$
- Let $i \in \{1, ..., m\}$. The $i$-th basic variable is $x_{ki}$
- Value of $x_{ki}$ as a function of $t$ is the $i$-th component of $x_B(t)$: $\beta_i - t\alpha_q^i$, where $\beta_i$ is the $i$-th component of $\beta$ and $\alpha_q^i$ is the $i$-th component of $\alpha_q$
- We need $\beta_i - t\alpha_q^i \geq 0 \iff \beta_i \geq t\alpha_q^i$
  - If $\alpha_q^i \leq 0$ the constraint is satisfied for all $t \geq 0$
  - If $\alpha_q^i > 0$ we need $\frac{\beta_i}{\alpha_q^i} \geq t$
- The best improvement is achieved with the strongest of the upper bounds:
  $$\theta := \min\{\frac{\beta_i}{\alpha_q^i} \mid \alpha_q^i > 0\}$$
- We say the $p$-th basic variable $x_{kp}$ is blocking or tight when $\theta = \frac{\beta_p}{\alpha_q^p}$.
  Then $\alpha_q^p$ is the pivot
Improving a Non-Optimal Solution

1. If $\theta = +\infty$
   (there is no upper bound, i.e., no $i$ such that $1 \leq i \leq m$ and $\alpha_q^i > 0$):

   Value of objective function can be decreased infinitely.
   LP is unbounded.
Improving a Non-Optimal Solution

1. If $\theta = +\infty$
   (there is no upper bound, i.e., no $i$ such that $1 \leq i \leq m$ and $\alpha^i_q > 0$):

   Value of objective function can be decreased infinitely. LP is **unbounded**.

2. If $\theta < +\infty$ and the $p$-th basic variable $x_{kp}$ is blocking:

   When setting $x_q = \theta$, the non-negativity of basic variables is respected.

   In particular the value of $x_{kp}$, i.e. the $p$-th component of $x_B(t)$, is
   \[ \beta_p - \theta \alpha^p_q = 0 \]

   We can make a **basis change**:
   $x_q$ enters the basis and $x_{kp}$ leaves, where $B = (k_1, \ldots, k_m)$.
Update

- **New basic indices:** $\bar{B} = (k_1, \ldots, k_{p-1}, q, k_{p+1} \ldots, k_m)$

  Before the $p$-th basic variable was $x_{k_p}$, now it is $x_q$

- **New basis:** $\bar{B} = B + (a_q - a_{k_p})e^T_p$

  where $e^T_p = (0, \ldots, 0, 1, 0, \ldots, 0)$ is the $p$-th unit vector

  The $p$-th column of the basis (which was $a_{k_p}$) is replaced by $a_q$.

- **New basic solution:** $\bar{\beta}_p = \theta$, $\bar{\beta}_i = \beta_i - \theta \alpha^i_q$ if $i \neq p$

  Note that before the $p$-th component of $\beta$ corresponded to $x_{k_p}$, now to $x_q$

- **New objective value:** $\bar{z} = z + \theta d_q$
Algorithmic Description

1. **Initialization:** Find an initial feasible basis $B$
   Compute $B^{-1}, \beta = B^{-1}b, z = c^T_B \beta$

2. **Pricing:** Compute $\pi^T = c^T_B B^{-1}$ and $d_j = c_j - \pi^T a_j$.
   If for all $j \in \mathcal{R}, d_j \geq 0$ then return **OPTIMAL**
   Else let $q$ be such that $d_q < 0$. Compute $\alpha_q = B^{-1} a_q$

3. **Ratio test:** Compute $\mathcal{I} = \{i \mid 1 \leq i \leq m, \alpha^i_q > 0\}$.
   If $\mathcal{I} = \emptyset$ then return **UNBOUNDED**
   Else compute $\theta = \min_{i \in \mathcal{I}} (\frac{\beta^i}{\alpha^i_q})$ and $p$ such that $\theta = \frac{\beta_p}{\alpha_q^p}$

4. **Update:**
   \[
   \bar{B} = B - \{k_p\} \cup \{q\} \quad \bar{B} = B + (a_q - a_{k_p}) e_p^T \\
   \bar{\beta}_p = \theta, \quad \bar{\beta}_i = \beta_i - \theta \alpha^i_q \quad \text{if } i \neq p \quad \bar{z} = z + \theta d_q
   \]
   Go to 2.
Updating Matrix Inverse

- Actually what we really care about is $B^{-1}$, not $B$
  We need it for computing $\pi = c_B^T B^{-1}$ and $\alpha_q = B^{-1}a_q$ at each step
  (and also $\beta = B^{-1}b$ in the initialization)

- Recomputing $B^{-1}$ at each iteration is too expensive
  (e.g. $O(m^3)$ arithmetic operations with Gaussian elimination!)

- Next slides: a more efficient way of computing $\bar{B}^{-1}$ using $B^{-1}$
Let us make a diversion into linear algebra

Let $b_1, ..., b_m$ be the columns of an invertible matrix $B$

Let $a, \alpha$ be such that $a = B\alpha = \sum_{i=1}^{m} \alpha_i b_i$

Let $p$ be such that $1 \leq p \leq m$

$B_a = (b_1, \ldots, b_{p-1}, a, b_{p+1}, \ldots, b_m)$. Want to compute $B_a^{-1}$
Let us make a diversion into linear algebra

Let \( b_1, \ldots, b_m \) be the columns of an invertible matrix \( B \)

Let \( a, \alpha \) be such that
\[
a = B\alpha = \sum_{i=1}^{m} \alpha_i b_i
\]

Let \( p \) be such that \( 1 \leq p \leq m \)

\( B_a = (b_1, \ldots, b_{p-1}, a, b_{p+1}, \ldots, b_m) \). Want to compute \( B_a^{-1} \)

Note \( \alpha_p \neq 0 \) as otherwise \( \text{rank}(B_a) < m \).

Then
\[
a = \alpha_p b_p + \sum_{i \neq p} \alpha_i b_i \quad \Rightarrow \quad b_p = \left(\frac{1}{\alpha_p}\right) a + \sum_{i \neq p} \left(\frac{-\alpha_i}{\alpha_p}\right) b_i
\]

Let \( \eta^T = \left( \left(\frac{-\alpha_1}{\alpha_p}\right), \ldots, \left(\frac{-\alpha_{p-1}}{\alpha_p}\right), \frac{1}{\alpha_p}, \left(\frac{-\alpha_{p+1}}{\alpha_p}\right), \ldots, \left(\frac{-\alpha_m}{\alpha_p}\right) \right)^T \).

Then
\[
b_p = B_a \eta
\]

Let \( E = (e_1, \ldots, e_{p-1}, \eta, e_{p+1}, \ldots, e_m) \)

where \( e_q \) is the \( q \)-th unit vector for \( 1 \leq q \leq m \).

Then
\[
B_a E = B \quad \Rightarrow \quad E^{-1} B_a^{-1} = B^{-1} \quad \Rightarrow \quad B_a^{-1} = E B^{-1}
\]
Updating Matrix Inverse

- Application to the simplex algorithm:
  \[ a = a_q, \alpha = \alpha_q, \text{ where } x_q \text{ is entering variable} \]
  Thus to update the inverse we can reuse already computed data!

- Using this update: \( B^{-1} \) is not actually represented as a square table, but as follows

- Assume initial basis is \( B_0 \) (e.g., unit matrix \( I \)).
  Then at the \( k \)-th iteration of the simplex algorithm the inverse matrix is
  \[ B^{-1} = E_k E_{k-1} \cdots E_2 E_1 B_0^{-1}, \]
  where \( E_i \) is the \( E \) matrix of the \( i \)-th iteration

- Each \( E \) matrix can be stored compactly (vector \( \eta \) + column index \( p \))

- We can represent \( B^{-1} \) as the list \( E_k, E_{k-1}, \ldots, E_2, E_1, B_0^{-1} \):
  Product Form of the Inverse (PFI)

- When the list is long we reset: the inverse is computed (reinversion)

- Other ways of representing \( B^{-1} \): LU factoriztion
Tableau vs. Revised Simplex

- **Time is saved:**
  - ✗ Tableau: all $d_k$, all $\alpha_k$ are computed
  - ✓ Revised: no. of non-basic variables $x_k$ for which $d_k$, $\alpha_k$ are computed can be adjusted

- **Space is saved:**
  - ✗ Tableau: even if $A$ sparse, tableau tends to get filled
  - ✓ Revised: sparsity of $A$ can be exploited for storage, and pivots can be chosen to represent $B^{-1}$ compactly

- **Better numerical behaviour:**
  - ✗ Tableau: errors due to floating-point arithmetic accumulate at each pivoting step
  - ✓ Revised: reinversion (PFI representation of $B^{-1}$) or refactorization (LU representation of $B^{-1}$) can be used for resetting