Further analysis of the (primal) simplex method

- Initializing the primal simplex method
- Computational efficiency of the simplex method
  - Implementation of a pivot (an iteration)
  - Total number of iterations required

How to find a starting BFS (or diagnose infeasibility)?

- Easy if LP has form $\min \{ c^T x : Ax \leq b, \ x \geq 0 \}$ with $b \geq 0$:
  $\min c^T x + 0^T z$
  s.t. $Ax + l z = b$
  $x, \ z \geq 0$

  $x = 0, \ z = b$ is an obvious starting BFS with $B = I$

- For LP $\min \{ c^T x : Ax = b, \ x \geq 0 \}$ with $b \geq 0$, consider an auxiliary problem with artificial variables $y$:
  $\begin{align*}
  (AUX) \min & \quad 0^T x + e^T y \\
  \text{s.t.} & \quad Ax + l y = b \\
  & \quad x, \ y \geq 0
  \end{align*}$

  $\triangleright$ LP is feasible iff optimal cost of (AUX) is 0
  $\triangleright$ $x = 0, \ y = b$ is an obvious starting BFS with $B = I$; solve with primal simplex method
  $\triangleright$ If in the optimal BSF of (AUX) found by simplex all $y_i$’s are non-basic, we have a BFS of the original LP
Driving artificial variables out of the basis

- Suppose value of (AUX) is 0, but some $y_i$'s are still basic at optimality, e.g.,

$$B = \begin{bmatrix} A_{B(1)} & A_{B(2)} & A_{B(3)} & A_{B(4)} & e_2 & e_7 & e_4 & e_3 \end{bmatrix}$$

- Suppose, wolog, $x_{B(1)}, \ldots, x_{B(k)}$ with $k < m$ are basic, and $(k + 1)$st through $m$th basic variables are $y_i$'s

- If $\exists j \notin B$ be such that $(B^{-1}A_j)_{k+1} \neq 0$, then
  - **Claim:** $A_j$ is linearly independent from $A_{B(1)}, \ldots, A_{B(k)}$
  - Change the basis: $x_j$ enters, $y_{B(k+1)}$ leaves
  - We are still at the same BFS, since $x_j = y_{B(k+1)} = 0$, but now the basis has one more “real” variable; repeat until all $y_i$'s are out of the basis

- What if the entire $(k + 1)$st row of $B^{-1}A$ is 0?
  - Then $(k + 1)$st constraints is redundant, and can be eliminated along with corresponding artificial variable

Two-phase simplex method

**Phase I:**

- Convert the LP into an equivalent one in standard form
- If necessary, multiply constraints by $-1$ so that $b \geq 0$
- Introduce artificial variables $y_1, \ldots, y_m$ and apply primal simplex method (with appropriate anti-cycling rule) to the auxiliary problem
- If optimal value of aux. problem is positive, stop: LP is infeasible
- Otherwise, LP is feasible. Use the above technique to
  - Eliminate redundant (linearly dependent) constraints from LP
  - Obtains a starting primal-feasible basis for LP (whose $A$ now has full row rank)

**Phase II:**

- Let the final basis obtained in Phase I be the initial basis for LP
- Apply primal simplex method (with appropriate anti-cycling rule) to LP
Algorithms and operation counts: Big-O notation

Comparing running time of algorithms:

- Running time may depend on programming and hardware
- Adequate approximation is provided by the number of arithmetic operations required by the algorithm

**Definition 1.2 (a)**

Let \( f \) and \( g \) be functions that map positive numbers to positive numbers. We say that \( f(n) = O(g(n)) \) if

\[
\exists n_0 > 0, \exists c > 0 : f(n) \leq cg(n) \forall n \geq n_0.
\]

- \( 4n^2 + 5n = O(n^2) \)
- Suppose \( n \geq m \)
  - \( O(m + n) = O(n) \); \( O(m^3 + mn + m + n) = O(m^3 + mn) \)
- Inner product of two \( n \)-vectors: \( O(n) \) operations
- Solving an \( n \times n \) system of linear equations: \( O(n^3) \) operations
- All of the above — polynomial rates of growth, compare with exponential: \( O(2^n) \)

Recall: An iteration of the (primal) simplex method

1. Let \( B \) be a primal feasible basis
2. Compute \( \bar{c}_j = c_j - c_B^T B^{-1} A_j \forall j \in N. \) If \( \bar{c} \geq 0 \) — terminate; optimal solution found. Else, choose \( j : \bar{c}_j < 0. \)
3. Let \( d \) be the \( j \) basic direction at \( x: \)

\[
d_i = 0, \ i \in N, \ i \neq j, \ d_j = 1, \ d_B = (d_B(1), \ldots, d_B(m))^T = -B^{-1}A_j
\]

If \( d_B \geq 0 \), terminate; LP is unbounded.
4. Else, let

\[
\theta^* = \min_{i : d_B(i) < 0} \frac{x_B(i)}{-d_B(i)}, \ I = \arg\min_{i : d_B(i) < 0} \frac{x_B(i)}{-d_B(i)}
\]

5. Let \( l \) be as above, \( \theta^* = \frac{x_B(l)}{-d_B(l)} \). Form a new basis by replacing \( B(l) \) with \( j. \)
### Implementation of the Simplex Method

**Naive:** no additional information retained/updated from iteration to iteration

- At the beginning of the iteration have indices $B(1), \ldots, B(m)$
- Form $B$ and compute $p^T = c_B^T B^{-1}$ (by solving system $B^T p = c_B$ for $p$) \((O(m^3))\) operations
- Compute reduced costs $\bar{c}^T = c^T - p^T A$ \((\text{between } O(m^2) \text{ and } O(nm))\) operations
  - Depending on the pivoting rule used, may compute them one at a time, or may have to compute all
- Select $A_j$ to enter the basis and compute $u = B^{-1} A_j$ (by solving $Bu = A_j$) \((O(m^3))\) operations
- Determine $\theta^*$, find the variable $l$ leaving the basis and update the basic indices \((O(m))\) operations

- Memory requirements: $O(m)$
- Computational effort: $O(m^3 + mn)$
  - Except for special cases with very simple structure of $B$, e.g., network flow problems

### Implementation of the Simplex Method

**Full Tableau Method**

Maintains a full representation of the LP *with respect to the current basis*:

\[
\min 0^T x_B + (c_N^T - c_B^T B^{-1} A_N) x_N + c_B^T B^{-1} b \\
\text{s.t.} \quad I x_B + B^{-1} A_N x_N = B^{-1} b \\
\quad x \geq 0
\]

- At the beginning of each iteration have:

\[
\begin{pmatrix}
-c_B^T B^{-1} b & 0^T & c_N^T - c_B^T B^{-1} A_N \\
B^{-1} b & I & B^{-1} A_N
\end{pmatrix}
\]

- Select $j$ to enter the basis \((\text{no operations!})\)
- Determine $\theta^*$, find the variable $l$ leaving the basis \((O(m))\) operations
- Manipulate the tableau to update it for the new basis \((O(mn))\) operations

- Memory requirements: $O(mn)$
- Computational effort: $O(mn)$
Implementation of the simplex method
Revised: $B^{-1}$ retained/updated

- At the beginning of the iteration have basic columns $A_{B(1)}, \ldots, A_{B(m)}$, associated BFS $x$, and $B^{-1}$
- Compute $p^T = c^T_B B^{-1}$ (by matrix-vector multiplication) $O(m^2)$
- Compute reduced costs $\bar{c}^T = c^T - p^T A$ $O(nm)$
- Select $A_j$ to enter the basis and compute $u = B^{-1} A_j$ $O(m^2)$
- Determine $\theta^*$, find the variable $B(l)$ leaving the basis and update the basic columns and corresponding BFS $O(m)$
- Compute $\bar{B}^{-1}$, where $\bar{B}$ is the new basis matrix, from $B^{-1}$ and $A_j$ $O(m^2)$
  - How? See the following slides
- Memory requirements: $O(m^2)$
- Computational effort: $O(m^2 + mn) = O(mn)$

Rank-1 Update Matrix: Sherman-Morrison Formula

- If $u, v \in \mathbb{R}^m$, then $uv^T \in \mathbb{R}^{m \times m}$, where $(uv^T)_{ij} = u_i v_j$
  - Called a “rank-1 matrix” — for obvious reasons
- Let $M \in \mathbb{R}^{m \times m}$ be a matrix.
- Suppose that we know $M^{-1}$
- Let $\tilde{M} = M + uv^T$. We want to find $\tilde{M}^{-1}$
- The Sherman-Morrison Formula: $\tilde{M}$ is invertible if and only if $v^T M^{-1} u \neq -1$, in which case
  $$\tilde{M}^{-1} = \left[ \mathbf{I} - \frac{M^{-1} uv^T}{1 + v^T M^{-1} u} \right] M^{-1}$$
- Compute $q = M^{-1} u$ and $r^T = v^T M^{-1}$ $O(m^2)$
- Compute $\alpha = 1 + v^T M^{-1} u = 1 + v^T q$ $O(m)$
- Now $\tilde{M}^{-1} = M^{-1} - \frac{q r^T}{\alpha}$ $O(m^2)$
Back to updating the basis

\[ \mathbf{B} = \begin{bmatrix} \mathbf{A}_{B(1)} & \cdots & \mathbf{A}_{B(l-1)} & \mathbf{A}_{B(l)} & \mathbf{A}_{B(l+1)} & \cdots & \mathbf{A}_{B(m)} \end{bmatrix} \]

\[ \tilde{\mathbf{B}} = \begin{bmatrix} \mathbf{A}_{B(1)} & \cdots & \mathbf{A}_{B(l-1)} & \mathbf{A}_j & \mathbf{A}_{B(l+1)} & \cdots & \mathbf{A}_{B(m)} \end{bmatrix} \]

Note that

\[ \tilde{\mathbf{B}} = \mathbf{B} + (\mathbf{A}_j - \mathbf{A}_{B(l)}) \times (\mathbf{e}_l)^T \]

where \( \mathbf{e}_l \) is the \( l \)th unit vector

\[ \tilde{\mathbf{B}} = \mathbf{B} + \mathbf{u} \mathbf{v}^T \]

with

\[ \mathbf{u} = (\mathbf{A}_j - \mathbf{A}_{B(l)}) \quad \text{and} \quad \mathbf{v} = (\mathbf{e}_l) \]

— can use rank-1 update formula to compute \( \tilde{\mathbf{B}}^{-1} \)

Simplex method(s): Remaining unanswered question

- How many iterations of the simplex method are required to solve an LP instance?
  - Will discuss later, as part of general discussion of computational efficiency of algorithms and computational complexity of problems