Parametric programming

\[
\begin{align*}
\text{minimize} & \quad (c + \theta d)'x \\
\text{subject to} & \quad Ax = b \\
& \quad x \geq 0,
\end{align*}
\]

Solve for every value of \( \theta \)

Example:

\[
\begin{align*}
\text{minimize} & \quad (-3 + 2\theta)x_1 + (3 - \theta)x_2 + x_3 \\
\text{subject to} & \quad x_1 + 2x_2 - 3x_3 + x_4 = 5 \\
& \quad 2x_1 + x_2 - 4x_3 + x_5 = 7 \\
& \quad x \geq 0
\end{align*}
\]

Optimal cost:

\[
g(\theta) = \min_{i=1,\ldots,N} (c + \theta d)'x_i,
\]

\(x^1, \ldots, x^N\) are the extreme points of the feasible set

(Parametric) simplex tableau

\[
\begin{array}{c|cccccc}
0 & -3 + 2\theta & 3 - \theta & 1 & 0 & 0 \\
5 & 1 & 2 & -3 & 1 & 0 \\
7 & 2 & 1 & -4 & 0 & 1 \\
\end{array}
\]

- If \(-3 + 2\theta \geq 0\) and \(3 - \theta \geq 0\), all reduced costs are non-negative and we have an optimal basic feasible solution.

\[
g(\theta) = 0, \quad \frac{3}{2} \leq \theta \leq 3.
\]

- For \(\theta > 3\), have \(x_2\) enter the basis

- New tableau:

\[
\begin{array}{c|cccccc}
-7.5 + 2.5\theta & -4.5 + 2.5\theta & 0 & 5.5 - 1.5\theta & -1.5 + 0.5\theta & 0 \\
2.5 & 0.5 & 1 & -1.5 & 0.5 & 0 \\
4.5 & 1.5 & 0 & -2.5 & -0.5 & 1 \\
\end{array}
\]

- All reduced costs nonnegative if \(3 \leq \theta \leq 5.5/1.5\)

- Optimal cost

\[
g(\theta) = 7.5 - 2.5\theta, \quad 3 \leq \theta \leq \frac{5.5}{1.5}
\]

- For \(\theta > 5.5/1.5\), reduced cost of \(x_3\) is negative.

- No positive pivot element

- For \(\theta > 5.5/1.5\), \(g(\theta) = -\infty\)

- Proceed similarly for \(\theta < 3/2\)
Parametric programming more generally

- Reduced costs depend linearly on $\theta$

- Bfs and basis matrix $B$, optimal for $\theta_1 \leq \theta \leq \theta_2$

- Reduced cost of $x_j$ negative for $\theta > \theta_2$.
  - Reduced cost is zero for $\theta = \theta_2$

- If $B^{-1}A_j \leq 0$, $g(\theta) = -\infty$ for $\theta > \theta_2$.

- Otherwise, bring $x_j$ into basis

- Still have optimal solution at $\theta = \theta_2$.
- Range of $\theta$ under which new basis is optimal $[\theta_2, \theta_3]$
- If $\theta_i < \theta_{i+1}$, no basis repeated twice
- Change of basis: breakpoints of $g(\theta)$
- If $\theta_i = \theta_{i+1}$, method may cycle

Dual parametric programming

- Keep $c$ fixed
- Right-hand side $b + \theta d$
- If increasing $\theta$ makes a basic variable negative, do a dual simplex iteration
Delayed column generation

\[ \begin{align*}
\text{minimize} & \quad c'x \\
\text{subject to} & \quad Ax = b \\
& \quad x \geq 0
\end{align*} \]

- \( A \) has a huge number of columns
  Can’t form \( A \) explicitly

- All that simplex needs is to discover \( i \) with \( r_i < 0 \) when one exists

- Assume we can solve the problem:
  \[ \text{minimize} \quad c_i - p'A_i \quad (= r_i) \]
  where \( p' = c'_pB^{-1} \)
  - Find \( j \) such that \( r_j \leq r_i \) for all \( i \)

- Run revised simplex
  - If \( r_j \geq 0 \), have optimal solution
  - If \( r_j < 0 \), \( A_j \) enters the basis

- Method terminates in the absence of degeneracy

Cutting stock problem

- Fabric rolls of width \( r \)
- Sizes of interest \( w_1, \ldots, w_m \)
  - Example: \( r = 10 \) and \( w_1 = 5, w_2 = 4, w_3 = 3 \).

- Demand \( b_i \) for each size \( w_i \)
- Minimize the number of rolls needed to satisfy demand
Cutting stock (ctd)

- Each roll is cut according to a certain pattern
- Example: \( r = 10 \) and \( w_1 = 5, w_2 = 4, w_3 = 3 \).
- Allowed patterns:
  \[
  A_1 = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \quad A_3 = \begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix}, \quad A_4 = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}
  \]
- A vector
  \[
  A_j = \begin{bmatrix} a_1 \\ \vdots \\ a_m \end{bmatrix}
  \]
is an allowed pattern if:
  \[
  \sum_{i=1}^{m} a_i w_i \leq r
  \]
  \( a_i \) integer, \( a_i \geq 0 \)
- Let \( x_j \) = number of rolls cut according to pattern \( A_j \)
  \[
  \text{minimize } \sum_{j} x_j \\
  \text{subject to } \sum_{j} A_j x_j = b \\
  x \geq 0
  \]

Cutting stock (ctd)

minimize \( \sum_{j} x_j \)
subject to \( \sum_{j} A_j x_j = b \)
\( x \geq 0 \)

- 1. Optimal solution need not be integer
- 2. Number of possible patterns is huge

- 1. Solve LP and round each \( x_j \) upwards
- 2. Use delayed column generation

- At each iteration, minimize \( \bar{c}_j = 1 - p'A_j \)
  - maximize \( p'A_j \)
  \[
  \text{maximize } \sum_{i=1}^{m} p_i a_i \\
  \text{subject to } \sum_{i=1}^{m} w_i a_i = r \\
  a_i \geq 0, \quad a_i \text{ integer}
  \]
- “Knapsack” problem (\( p_i \) =value, \( w_i \) =weight)
- Despite integrality constraints, can be solved fairly efficiently
Variant with retained columns

• Keep some columns $A_i, i \in I$, in memory
  (The basic columns plus, possibly, more)

• Look for $j$ with $\bar{r}_j < 0$
  – Look only inside the set $I$
  – Same as solving restricted problem:
    
    $$\text{minimize} \quad c^T x$$
    $$\text{subject to} \quad \sum_{i \in I} A_i x_i = b$$
    $$x \geq 0$$

• When at optimal of restricted problem,
  look outside the set $I$ for $j$ with $\bar{r}_j < 0$
• Form new set $I$ (that includes $j$) and restart

• Extreme variants:
  – $I_1$ = set of basic indices
  – $I_2$ = indices of all columns generated in the past
• All variants terminate under nondegeneracy

Cutting plane methods

• Dual of standard form problem:
  
  $$\text{maximize} \quad p^T b$$
  $$\text{subject to} \quad p^T A_i \leq c_i, \quad i = 1, \ldots, n,$$

• Large number $n$ of constraints
• Let $I \subset \{1, \ldots, n\}$
• Solve relaxed dual problem
  
  $$\text{maximize} \quad p^T b$$
  $$\text{subject to} \quad p^T A_i \leq c_i, \quad i \in I,$$

• If optimal solution of relaxed problem
  satisfies all constraints of original problem,
  then it is optimal for the latter
Cutting planes (continued)

- If optimal solution of relaxed problem is infeasible, bring a violated constraint into $I$
- Method needs:
  - A way of checking feasibility
  - A way of identifying violated constraints
- One possibility
  \[ \text{minimize } c_i - (p^*)'A_i \]
- Cutting planes for dual = Column generation for primal
- Options:
  - Retain old constraints
  - Discard (some) inactive constraints